Acta Informatica (2017) 54:217–242
DOI 10.1007/s00236-016-0287-9

CrossMark

ORIGINAL ARTICLE

Dynamic Bayesian networks for formal verification of structured stochastic processes

Sadegh Esmaeil Zadeh Soudjani[1] · **Alessandro Abate**[1] · **Rupak Majumdar**[2]

Received: 13 December 2015 / Accepted: 15 November 2016 / Published online: 3 December 2016

Abstract We study the problem of finite-horizon probabilistic invariance for discrete-time Markov processes over general (uncountable) state spaces. We compute discrete-time, finite-state Markov chains as formal abstractions of the given Markov processes. Our abstraction differs from existing approaches in two ways: first, we exploit the structure of the underlying Markov process to compute the abstraction separately for each dimension; second, we employ dynamic Bayesian networks (DBN) as compact representations of the abstraction. In contrast, approaches which represent and store the (exponentially large) Markov chain explicitly incur significantly higher memory requirements. In our experiments, explicit representations scaled to models of dimension less than half the size as those analyzable by DBN representations. We show how to construct a DBN abstraction of a Markov process satisfying an independence assumption on the driving process noise. We compute a guaranteed bound on the error in the abstraction w.r.t. the probabilistic invariance property—the dimension-dependent abstraction makes the error bounds more precise than existing approaches. Additionally, we show how factor graphs and the sum-product algorithm for DBNs can be used to solve the finite-horizon probabilistic invariance problem. Together, DBN-based representations and algorithms can be significantly more efficient than explicit representations of Markov chains for abstracting and model checking structured Markov processes.

This work was partially supported by the European Commission IAPP project AMBI 324432, and by the John Fell OUP Research Fund. Part of the results of this paper has been presented at CONCUR 2015 [13].

✉ Sadegh Esmaeil Zadeh Soudjani
sadegh.soudjani@cs.ox.ac.uk

Alessandro Abate
alessandro.abate@cs.ox.ac.uk

Rupak Majumdar
rupak@mpi-sws.org

[1] Department of Computer Science, University of Oxford, Oxford, United Kingdom

[2] Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

1 Introduction

Markov processes over general (uncountable) state spaces appear in many areas of engineering, such as power and transportation networks, biological processes, robotics, and manufacturing systems. The importance of this class of stochastic processes in applications has motivated a significant research effort into their foundations, analysis, and verification.

We study the problem of algorithmically verifying finite-horizon probabilistic invariance for Markov processes, that is computing the probability that a stochastic process remains within a given set for a given finite time horizon. For finite-state stochastic processes, there is a mature theory of model checking discrete-time Markov chains [7], and a number of probabilistic model checking tools [18,22] compute explicit solutions to related verification problems. On the other hand, stochastic processes taking values over uncountable state spaces do not in general admit explicit solutions, and related verification problems are undecidable even for simple dynamics [2]. A number of studies have therefore explored *abstraction* techniques that reduce the given stochastic process (over a general state space) to a finite-state process, while preserving properties in a quantitative sense [2,10]. The abstracted model allows the application of standard model checking techniques (and software tools) over finite-state models. The work in [2] has further shown that an explicit error can be attached to the abstraction: this error is computed purely based on continuity properties of the concrete Markov process. As such, properties proved on the finite-state abstraction can be used to reason about properties of the original process. The overall approach has been customized under various assumptions on the model [9,11] and has been extended to linear temporal specifications [3,30]. A software tool has also been developed to automate the abstraction procedure [14] and to couple it with standard probabilistic model checkers [18,22].

In previous work, the structure of the underlying Markov process (namely, the interdependence among its variables) has not been actively reflected in the abstraction algorithms, and the finite-state Markov chain has been always represented explicitly, which can become quite expensive in terms of memory requirements. In many applications, the dynamics of the Markov process, which are fully characterized by a conditional stochastic kernel, often exhibit specific structural properties. More precisely, the dynamics of any state variable may depend only on a limited number of other state variables, and the process noise driving each state variable can be assumed to be independent. Examples of such structured systems are models of power grids and sensor–actuator networks as large-scale interconnected networks [29], and mass-spring-damper systems [5,6] with a given non-dense topology.

In this work we present an abstraction and model checking algorithm for discrete-time stochastic dynamical systems over general (uncountable) state spaces. The procedure constructs a finite-state Markov abstraction of the process, but differs from previous work in that it is based on a dimension-dependent partitioning of the state space. Additionally, we perform a precise dimension-dependent analysis of the error introduced by the abstraction, and our error bounds can be exponentially smaller than the earlier bounds obtained in [2]. Furthermore, we represent the abstraction as a dynamic Bayesian network (DBN) [19], instead of explicitly representing it via a probabilistic transition matrix. The Bayesian network representation exploits independence assumptions in the model to potentially provide polynomially sized representations (in the number of dimensions) for the Markov chain abstraction, whereas the explicit transition matrix would be exponential in the number of dimensions. We show how factor graphs and the sum-product algorithm, developed for belief propagation in Bayesian networks, can be used to model check probabilistic invariance properties without constructing the transition matrix. Overall, our approach leads to significant reduction in computational

and memory resources for model checking structured Markov processes, and provides tighter error bounds.

The material is organized in seven sections. Section 2 defines discrete-time Markov processes and the probabilistic invariance problem. Section 3 presents a new algorithm for abstracting a process to a DBN, together with the quantification of the abstraction error. We discuss efficient model checking of the constructed DBN in Sect. 4. The performance of the DBN abstraction approach is compared with the state-of-the-art abstraction procedure in Sect. 5. We apply the overall abstraction algorithm to a case study in Sect. 6. Section 7 outlines current directions of investigation.

2 Markov processes and probabilistic invariance

2.1 Discrete-time Markov processes

We write $\mathbb{N}$ for the non-negative integers $\mathbb{N} := \{0, 1, 2, \ldots\}$ and $\mathbb{N}_n$ for positive integers not greater than n, $\mathbb{N}_n := \{1, 2, \ldots, n\}$. We use bold typeset for vectors and normal typeset for one-dimensional quantities.

We consider discrete-time stochastic dynamical systems defined over a general state space $\mathcal{S}$. For a sequence of independent and identically distributed (iid) random variables $\{\boldsymbol{\zeta}(t),\ t \in \mathbb{N}\}$ taking values in $\mathbb{R}^n$, and a measurable map $\boldsymbol{f} : \mathcal{S} \times \mathbb{R}^n \to \mathcal{S}$, the dynamical system is characterized as

$$\boldsymbol{s}(t+1) = \boldsymbol{f}(\boldsymbol{s}(t), \boldsymbol{\zeta}(t)), \quad \forall t \in \mathbb{N}, \quad \boldsymbol{s}(0) = \boldsymbol{s}_0 \in \mathcal{S}. \tag{1}$$

The stochastic dynamical system (1) can be seen as a discrete-time Markov process $\mathscr{M}_{\mathfrak{s}}$ characterized by the tuple $(\mathcal{S}, \mathcal{B}, T_{\mathfrak{s}})$: $\mathcal{S}$ is the continuous state space, which we assume to be endowed with a metric and to be separable[1]; $\mathcal{B}$ is the Borel σ-algebra associated to $\mathcal{S}$, which is the smallest σ-algebra containing all open subsets of $\mathcal{S}$; and $T_{\mathfrak{s}} : \mathcal{S} \times \mathcal{B} \to [0, 1]$ is a stochastic kernel, so that $T_{\mathfrak{s}}(\cdot, B)$ is a non-negative measurable function for any set $B \in \mathcal{B}$, and $T_{\mathfrak{s}}(\boldsymbol{s}, \cdot)$ is a probability measure on $(\mathcal{S}, \mathcal{B})$ for any $\boldsymbol{s} \in \mathcal{S}$. The stochastic kernel $T_{\mathfrak{s}}(\boldsymbol{s}, \cdot)$ of dynamical system (1) is computed as

$$T_{\mathfrak{s}}(\boldsymbol{s}, B) = T_{\zeta}\left(\boldsymbol{\zeta} \in \mathbb{R}^n \,:\, \boldsymbol{f}(\boldsymbol{s}, \boldsymbol{\zeta}) \in B\right),$$

where T_{ζ} is the distribution of the r.v. $\boldsymbol{\zeta}(0)$ (in fact, of any $\boldsymbol{\zeta}(t)$ since these are iid random variables). In other words, the map $\boldsymbol{f}$ and the distribution of the r.v. $\{\boldsymbol{\zeta}(t)\}$ uniquely define the stochastic kernel of the process.

Trajectories (also called traces or paths) of $\mathscr{M}_{\mathfrak{s}}$ are sequences $(\boldsymbol{s}(0), \boldsymbol{s}(1), \boldsymbol{s}(2), \ldots)$ which belong to the set $\Omega = \mathcal{S}^{\mathbb{N}}$. The product σ-algebra on Ω is denoted by $\mathcal{F}$. Given the initial state $\boldsymbol{s}(0) = \boldsymbol{s}_0 \in \mathcal{S}$ of $\mathscr{M}_{\mathfrak{s}}$, the stochastic Kernel $T_{\mathfrak{s}}$ induces a unique probability measure $\mathcal{P}$ on $(\Omega, \mathcal{F})$ that satisfies the Markov property: namely for any measurable set $B \in \mathcal{B}$ and any $t \in \mathbb{N}$

$$\mathcal{P}\left(\boldsymbol{s}(t+1) \in B | \boldsymbol{s}(0), \boldsymbol{s}(1), \ldots, \boldsymbol{s}(t)\right) = \mathcal{P}\left(\boldsymbol{s}(t+1) \in B | \boldsymbol{s}(t)\right) = T_{\mathfrak{s}}(\boldsymbol{s}(t), B).$$

We assume that the stochastic kernel $T_{\mathfrak{s}}$ admits a density function $t_{\mathfrak{s}} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$, such that $T_{\mathfrak{s}}(\boldsymbol{s}, B) = \int_B t_{\mathfrak{s}}(\bar{\boldsymbol{s}}|\boldsymbol{s}) d\bar{\boldsymbol{s}}$.

[1] A metric space $\mathcal{S}$ is called separable if it admits a countable dense subset.

Let us expand the dynamical Eq. (1) explicitly over its states $\boldsymbol{s} = [s_1, \ldots, s_n]^T$, map components $\boldsymbol{f} = [f_1, \ldots, f_n]^T$, and uncertainly terms $\boldsymbol{\zeta} = [\zeta_1, \ldots, \zeta_n]^T$, as follows:

$$\begin{aligned} s_1(t+1) &= f_1(s_1(t), s_2(t), \ldots, s_n(t), \zeta_1(t)), \\ s_2(t+1) &= f_2(s_1(t), s_2(t), \ldots, s_n(t), \zeta_2(t)), \\ &\vdots \\ s_n(t+1) &= f_n(s_1(t), s_2(t), \ldots, s_n(t), \zeta_n(t)). \end{aligned} \tag{2}$$

In this article we are interested in exploiting the knowledge of the structure of the dynamics in (2), in order to scale up formal verification algorithms based on abstractions [2,10,11]. We focus our attention on continuous (unbounded and uncountable) Euclidean spaces $\mathcal{S} = \mathbb{R}^n$, and further assume that for any $t \in \mathbb{N}$, $\zeta_k(t)$ are independent for all $k \in \mathbb{N}_n$. This latter assumption is widely used in the theory of dynamical systems, and allows for the following multiplicative structure on the conditional density function of the process:

$$t_{\mathfrak{s}}(\bar{\boldsymbol{s}}|\boldsymbol{s}) = t_1(\bar{s}_1|\boldsymbol{s}) t_2(\bar{s}_2|\boldsymbol{s}) \ldots t_n(\bar{s}_n|\boldsymbol{s}), \tag{3}$$

where the function $t_k : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ solely depends on the map f_k and the distribution of ζ_k. The following example is adapted from [12] to demonstrate the computation of the function t_k based on some regularity assumptions on the function f_k.

Example 1 Consider a kth order version of the system of equations in (2),

$$s_k(t+1) = f_k(\boldsymbol{s}(t), \zeta_k(t)), \quad \boldsymbol{s}(\cdot) \in \mathbb{R}^n, \ \ \zeta_k(\cdot) \in \mathbb{R},$$

where $\zeta_k(\cdot)$ are iid with known distribution $t_{\zeta_k}(\cdot)$. Suppose that the vector field $f_k : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is continuously differentiable and that $\frac{\partial f_k}{\partial \zeta_k}$ is invertible. Then the *implicit function theorem* guarantees the existence and uniqueness of a function $g_k : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$ such that $\zeta_k(t) = g_k(s_k(t+1), \boldsymbol{s}(t))$. The conditional density function t_k in this case is [27]:

$$t_k(\bar{s}_k|\boldsymbol{s}) = \left| \det \left[\frac{\partial g_k}{\partial \bar{s}_k}(\bar{s}_k, \boldsymbol{s}) \right] \right| t_{\zeta_k}(g_k(\bar{s}_k, \boldsymbol{s})).$$

As a special case the invertibility of $\frac{\partial f_k}{\partial \zeta_k}$ is guaranteed for systems with additive process noise, namely $f_k(\boldsymbol{s}, \zeta_k) = f_{kd}(\boldsymbol{s}) + \zeta_k$, which results in $t_k(\bar{s}_k|\boldsymbol{s}) = t_{\zeta_k}(\bar{s}_k - f_{kd}(\boldsymbol{s}))$. This fact is used in the subsequent examples and in Sect. 5. □

Remark 1 The results of this article are presented under the structural assumption that $\zeta_k(\cdot)$ are independent over $k \in \mathbb{N}_n$. These results can be generalized to a broader class of processes by allowing inter-dependencies between the entries of the process noise, which leads to form subsets of the entries of $\boldsymbol{\zeta}(\cdot)$, which are so that any two entries from different subsets are independent, whereas entries within a subset may be dependent. This assumption induces a multiplicative structure on $t_{\mathfrak{s}}(\bar{\boldsymbol{s}}|\boldsymbol{s})$ among the different subsets, which is similar to (3). As it will be discussed in Sect. 3, our abstraction approach requires partitioning the state space projected over these independent subsets, thus algorithmically the higher the number of subsets, the more efficient our abstraction process. □

The following two examples provide instances of stochastic dynamical systems (2) and justify the structural assumption raised in (3).

Example 2 Figure 1 shows a system of n masses connected by springs and dampers. For $i \in \mathbb{N}_n$, block i has mass m_i, the ith spring has stiffness k_i, and the ith damper has damping coefficient b_i. The first mass is connected to a fixed wall by the left-most spring/damper

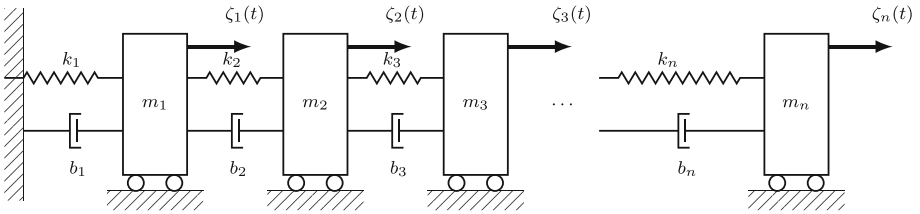

Fig. 1 An n-body mass-spring-damper system

connection. All other masses are connected to the previous mass with a spring and a damper. A force ζ_i is applied to each mass, modeling the effect of a disturbance or of process noise. The dynamics of the overall system is comprised of the position and velocity of the blocks. It can be shown that the dynamics in discrete time take the form $\boldsymbol{s}(t+1) = \Phi \boldsymbol{s}(t) + \boldsymbol{\zeta}(t)$, where $\boldsymbol{s}(t) \in \mathbb{R}^{2n}$ with $s_{2i-1}(t)$, $s_{2i}(t)$ indicating the velocity and position of mass i. The state transition matrix $\Phi = [\Phi_{ij}]_{i,j} \in \mathbb{R}^{2n \times 2n}$ is a band matrix with lower and upper bandwidth 3 and 2, respectively ($\Phi_{ij} = 0$ for $j < i - 3$ and for $j > i + 2$). □

Example 3 A second example of structured dynamical systems is a discrete-time large-scale interconnected system. Consider an interconnected system of $N_{\mathfrak{d}}$ heterogeneous linear time-invariant (LTI) subsystems described by the following stochastic difference equations:

$$\boldsymbol{s}_i(t+1) = \Phi_i \boldsymbol{s}_i(t) + \sum_{j \in N_i} G_{ij} \boldsymbol{s}_j(t) + B_i \boldsymbol{u}_i(t) + \boldsymbol{\zeta}_i(t),$$

where $i \in \mathbb{N}_{N_{\mathfrak{d}}}$ denotes the i^{th} subsystem and $\boldsymbol{s}_i \in \mathbb{R}^{n \times 1}$, $\boldsymbol{u}_i \in \mathbb{R}^{p \times 1}$, $\boldsymbol{\zeta}_i \in \mathbb{R}^{m \times 1}$ are the state, the input, and the process noise of subsystem i. The term $\sum_{j \in N_i} G_{ij} \boldsymbol{s}_j(t)$ represents the physical interconnection between the subsystems where N_i, $|N_i| \ll N_{\mathfrak{d}}$, is the set of subsystems to which system i is physically connected. The described interconnected system can be found in many application areas including smart power grids, traffic systems, and sensor-actuator networks [16]. □

2.2 Probabilistic invariance

We focus on verifying probabilistic invariance, which plays a central role in verifying properties of a system expressed as PCTL formulae or as linear temporal specifications [3,7,28,30].

Definition 1 (*Probabilistic invariance*) Consider a bounded Borel set $A \in \mathcal{B}$, representing a set of safe states. The finite-horizon *probabilistic invariance problem* asks to compute the probability that a trajectory of $\mathscr{M}_{\mathfrak{s}}$ associated with an initial condition $\boldsymbol{s}_0$ remains within the set A during the finite time horizon N:

$$p_N(\boldsymbol{s}_0, A) = \mathcal{P}\{\boldsymbol{s}(t) \in A \text{ for all } t = 0, 1, 2, \ldots, N | \boldsymbol{s}(0) = \boldsymbol{s}_0\}.$$

This quantity allows us to extend the result to a general probability distribution $\pi : \mathcal{B} \to [0, 1]$ for the initial state $\boldsymbol{s}(0)$ of the system as

$$\mathcal{P}\{\boldsymbol{s}(t) \in A \text{ for all } t = 0, 1, 2, \ldots, N\} = \int_{\mathcal{S}} p_N(\boldsymbol{s}_0, A) \pi(d\boldsymbol{s}_0). \tag{4}$$

Solution of the probabilistic invariance problem can be characterized via the value functions $V_k : \mathcal{S} \to [0, 1]$, $k = 0, 1, 2, \ldots, N$, defined by the following Bellman backward recursion [2]:

$$V_k(\boldsymbol{s}) = \mathbf{1}_A(\boldsymbol{s}) \int_A V_{k+1}(\bar{\boldsymbol{s}}) t_{\mathfrak{s}}(\bar{\boldsymbol{s}}|\boldsymbol{s}) d\bar{\boldsymbol{s}} \quad \text{for} \quad k = 0, 1, 2, \ldots, N-1. \tag{5}$$

This recursion is initialized with $V_N(\boldsymbol{s}) = \mathbf{1}_A(\boldsymbol{s})$, where $\mathbf{1}_A(\boldsymbol{s})$ is the indicator function which is 1 if $\boldsymbol{s} \in A$ and 0 otherwise, and results in the solution $p_N(\boldsymbol{s}_0, A) = V_0(\boldsymbol{s}_0)$.

Equation (5) characterizes the finite-horizon probabilistic invariance quantity as the solution of a dynamic programming problem. However, since its explicit solution is in general not available, the actual computation of the quantity $p_N(\boldsymbol{s}_0, A)$ requires N numerical integrations at each state in the set A. This is usually performed with techniques based on state-space discretization [8].

3 Formal abstractions as dynamic Bayesian networks

3.1 Dynamic Bayesian networks

A Bayesian network (BN) is a tuple $\mathfrak{B} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$. The pair $(\mathcal{V}, \mathcal{E})$ is a directed acyclic graph (DAG) representing the structure of the network. The nodes in $\mathcal{V}$ are (discrete or continuous) random variables and the arcs in $\mathcal{E}$ represent the dependence relationships among the random variables. The set $\mathcal{T}$ contains conditional probability distributions (CPD) in forms of tables or density functions for discrete and continuous random variables, respectively. In a BN, knowledge is represented in two ways: qualitatively, as dependences between variables by means of the DAG; and quantitatively, as conditional probability distributions attached to the dependence relationships. Each random variable $X_i \in \mathcal{V}$ is associated with a conditional probability distribution $\mathbb{P}(X_i|Pa(X_i))$, where $Pa(Y)$ represents the parent set of the variable $Y \in \mathcal{V}$: $Pa(Y) = \{X \in \mathcal{V} | (X, Y) \in \mathcal{E}\}$. A BN is called *two-layered* if the set of nodes $\mathcal{V}$ can be partitioned to two sets $\mathcal{V}_1, \mathcal{V}_2$ with the same cardinality such that only the nodes in the second layer $\mathcal{V}_2$ have an associated CPD.

A dynamic Bayesian network (DBN) [19,25] is a way to extend Bayesian networks to model probability distributions over collections of random variables $X(0), X(1), X(2), \ldots$ indexed by time t. A DBN[2] is defined to be a pair $(\mathfrak{B}_0, \mathfrak{B}_\rightarrow)$, where $\mathfrak{B}_0$ is a BN which defines the distribution of $X(0)$, and $\mathfrak{B}_\rightarrow$ is a two-layered BN that defines the transition probability distribution for $(X(t+1)|X(t))$.

3.2 DBNs as representations of Markov processes

We now show that any discrete-time Markov process $\mathscr{M}_{\mathfrak{s}}$ over $\mathbb{R}^n$ can be represented as a DBN $(\mathfrak{B}_0, \mathfrak{B}_\rightarrow)$ over n continuous random variables. The advantage of the reformulation is that it makes the dependencies between random variables explicit.

The BN $\mathfrak{B}_0$ is trivial for a given initial state of the Markov process $\boldsymbol{s}(0) = \boldsymbol{s}_0$. The DAG of $\mathfrak{B}_0$ has the set of nodes $\{X_1, X_2, \ldots, X_n\}$ without any arc. The Dirac delta distribution located in the initial state of the process is assigned to each node of $\mathfrak{B}_0$.[3] The DAG for the two-layered BN $\mathfrak{B}_\rightarrow = (\mathcal{V}, \mathcal{E}, \mathcal{T})$ comprises a set of nodes $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, with $\mathcal{V}_1 = \{X_1, X_2, \ldots, X_n\}$ and $\mathcal{V}_2 = \{\bar{X}_1, \bar{X}_2, \ldots, \bar{X}_n\}$. Each arc in $\mathcal{E}$ connects a node in $\mathcal{V}_1$ to another node in $\mathcal{V}_2$;

[2] The DBNs considered in this paper are stationary (the structure of the network does not change with the time index t). They have no input variables and are fully observable: the output of the DBN model is its entire state.

[3] For a general initial probability distribution $\pi : \mathcal{B} \rightarrow [0, 1]$, a set of arcs must be added to reflect its possible product structure. This construction is not important at the current stage because of the backward recursion formulation of the probabilistic safety (please refer to (4) in Sect. 2.2).

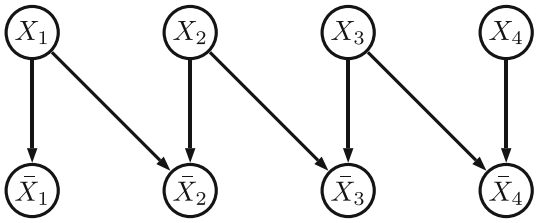

Fig. 2 Two-layered BN $\mathfrak{B}_{\rightarrow}$ associated with the stochastic linear dynamical system in (7) for $n = 4$

$(X_i, \bar{X}_j) \in \mathcal{E}$ if and only if $t_j(\bar{s}_j|\boldsymbol{s})$ is not a constant function of s_i. The set $\mathcal{T}$ assigns a CPD to each node $\bar{X}_j$ according to the density function $t_j(\bar{s}_j|\boldsymbol{s})$.

Example 4 Consider the following stochastic linear dynamical system:

$$\begin{aligned} s_1(t+1) &= a_{11}s_1(t) + \zeta_1(t) \\ s_2(t+1) &= a_{21}s_1(t) + a_{22}s_2(t) + \zeta_2(t) \\ s_3(t+1) &= a_{32}s_2(t) + a_{33}s_3(t) + \zeta_3(t) \\ &\vdots \\ s_n(t+1) &= a_{n(n-1)}s_{n-1}(t) + a_{nn}s_n(t) + \zeta_n(t), \end{aligned} \tag{6}$$

with initial state $\boldsymbol{s}(0) = \boldsymbol{s}_0 = [s_{01}, s_{02}, \ldots, s_{0n}]^T$, where $\zeta_i(\cdot)$, $i \in \mathbb{N}_n$ are independent Gaussian r.v. $\mathcal{N}(0, \sigma_i^2)$, which clearly satisfies the independence assumption on the process noise raised in Sect. 2.1. The conditional density function of the system takes the following form:

$$t_{\mathfrak{s}}(\bar{\boldsymbol{s}}|\boldsymbol{s}) = t_1(\bar{s}_1|s_1)t_2(\bar{s}_2|s_1, s_2)t_3(\bar{s}_3|s_2, s_3)\ldots t_n(\bar{s}_n|s_{n-1}, s_n).$$

The DAG of the two-layered BN $\mathfrak{B}_{\rightarrow}$ associated with this system is sketched in Fig. 2 for $n = 4$. The BN $\mathfrak{B}_0$ has an empty graph on the set of nodes $\{X_1, \ldots, X_n\}$ with the associated Dirac delta density functions located at s_{0i}, $\delta_d(s_i(0) - s_{0i})$.

Note that model (6) is in the form

$$\boldsymbol{s}(t+1) = \Phi\boldsymbol{s}(t) + \boldsymbol{\zeta}(t) \quad t \in \mathbb{N}, \tag{7}$$

for a lower bidiagonal matrix $\Phi = [a_{ij}]_{i,j}$ and independent Gaussian r.v. $\boldsymbol{\zeta}(t) \sim \mathcal{N}(0, \Sigma)$ with the diagonal covariance matrix $\Sigma = diag([\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2])$. For the linear dynamical system (7), which has a non-diagonal covariance matrix Σ, a linear transformation can be employed to change the coordinates and to obtain a stochastic linear system with a diagonal covariance matrix satisfying the independence assumption on the process noise raised in Sect. 2.1. □

3.3 Finite abstraction of Markov processes as discrete DBNs

Let $A \in \mathcal{B}$ be a bounded Borel set of safe states. We abstract the structured Markov process $\mathscr{M}_{\mathfrak{s}}$ interpreted in the previous section as a DBN with continuous variables to a DBN with discrete random variables. Our abstraction is relative to the set A. Algorithm 1 provides the steps of the abstraction procedure. It consists of discretizing each dimension into a finite number of bins.

The first step of Algorithm 1 is to project the safe set A over different dimensions, $D_i \doteq \Pi_i(A)$, where the projection operators $\Pi_i : \mathbb{R}^n \to \mathbb{R}$, $i \in \mathbb{N}_n$, are defined as $\Pi_i(\boldsymbol{s}) = s_i$ for any $\boldsymbol{s} = [s_1, \ldots, s_n]^T \in \mathbb{R}^n$. In step 2 of the Algorithm, set D_i is partitioned as $\{D_{ij}\}_{j=1}^{n_i}$ (for

Algorithm 1 Abstraction of model $\mathscr{M}_{\mathfrak{s}}$ as a DBN with $\mathfrak{B}_{\rightarrow} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$ over discrete r.v.

Require: input model $\mathscr{M}_{\mathfrak{s}} = (\mathcal{S}, \mathcal{B}, T_{\mathfrak{s}})$, safe set A

1: Project safe set A in each dimension $D_i \doteq \Pi_i(A),\ i \in \mathbb{N}_n$

2: Select finite n_i-dimensional partition of D_i as $D_i = \cup_{j=1}^{n_i} D_{ij},\ i \in \mathbb{N}_n$

3: For each D_{ij}, select single representative point $z_{ij} \in D_{ij},\ z_{ij} = \xi_i(D_{ij})$

4: Construct the DAG $(\mathcal{V}, \mathcal{E})$, with $\mathcal{V} = \{X_i, \bar{X}_i, i \in \mathbb{N}_n\}$ and $\mathcal{E}$ as per Sect. 3.2

5: Define $Z_i = \{z_{i1}, \ldots, z_{in_i}\}, i \in \mathbb{N}_n$, and take $\Omega_i = Z_i \cup \{\phi_i\}$ as the finite state space of two r.v. X_i and $\bar{X}_i$, ϕ_i being dummy states as per Sect. 3.3

6: Compute elements of the set $\mathcal{T}$, namely CPD T_i related to the node $\bar{X}_i, i \in \mathbb{N}_i$, as

$$T_i(\bar{X}_i = z | v(Pa(\bar{X}_i))) = \begin{cases} \int_{\Xi_i(z)} t_i(\bar{s}_i | v(Pa(\bar{X}_i)))d\bar{s}_i, & z \in Z_i,\ v(Pa(\bar{X}_i)) \cap \phi = \emptyset \\ 1 - \sum\limits_{z \in Z_i} \int_{\Xi_i(z)} t_i(\bar{s}_i | v(Pa(\bar{X}_i)))d\bar{s}_i, & z = \phi_i,\ v(Pa(\bar{X}_i)) \cap \phi = \emptyset \\ 1, & z = \phi_i,\ v(Pa(\bar{X}_i)) \cap \phi \neq \emptyset \\ 0, & z \in Z_i,\ v(Pa(\bar{X}_i)) \cap \phi \neq \emptyset \end{cases}$$

Ensure: output DBN with $\mathfrak{B}_{\rightarrow} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$ over discrete r.v.

any $i \in \mathbb{N}_n$, D_{ij}'s are arbitrary but non-empty, non-intersecting, and $D_i = \cup_{j=1}^{n_i} D_{ij}$). In the next step, representative points $z_{ij} \in D_{ij}$ are also chosen arbitrarily. The subsequent results are independent of the choice of these representative points, but a natural option for interval partition sets D_{ij} is their center point. Then the DAG $(\mathcal{V}, \mathcal{E})$ of the DBN $\mathfrak{B}_{\rightarrow}$ is constructed with $\mathcal{V} = \{X_i, \bar{X}_i, i \in \mathbb{N}_n\}$ and $\mathcal{E}$ as per Sect. 3.2. Step 5 of the algorithm constructs the support of the random variables in $\mathcal{V}$. For any $i \in \mathbb{N}_n$, the support of $X_i, \bar{X}_i$ will be $\Omega_i \doteq Z_i \cup \{\phi_i\}$ with the set $Z_i \doteq \{z_{i1}, \ldots, z_{in_i}\}$ containing the representative points selected in step 3 and the dummy state ϕ_i representing the complement of the set D_i. Finally, step 6 computes the discrete CPDs $T_i(\bar{X}_i | Pa(\bar{X}_i))$, reflecting the dependencies among the variables.

Each row of the CPD T_i includes values of conditional random variables $Pa(\bar{X}_i)$, the value of r.v. $\bar{X}_i$, and their associated probability. This probability is written as $T_i(\bar{X}_i = z | v(Pa(\bar{X}_i)))$ in step 6 of the algorithm. In other words, the function $v(\cdot)$ acts on (possibly a set of) random variables and provides their instantiation. The term $v(Pa(\bar{X}_i))$ that is present in the conditioned argument of t_i leads to evaluate function $t_i(\bar{s}_i | \cdot)$ at the instantiated values of $Pa(\bar{X}_i)$.

For any $i \in \mathbb{N}_n$, $\Xi_i : Z_i \rightarrow 2^{D_i}$ represents a set-valued map that associates to any point $z_{ij} \in Z_i$ the corresponding partition set $D_{ij} \subset D_i$ (this is known as the "refinement map"). Furthermore, the abstraction map $\xi_i : D_i \rightarrow Z_i$ associates to any point $s_i \in D_i$ the corresponding discrete state in Z_i. Additionally, notice that the absorbing states $\phi = \{\phi_1, \ldots, \phi_n\}$ are added to the definition of BN $\mathfrak{B}_{\rightarrow}$ so that the conditional probabilities $T_i(\bar{X}_i | Pa(\bar{X}_i))$ marginalize to one.

The construction of the DBN with discrete r.v. in Algorithm 1 is closely related to the Markov chain abstraction method in [2,10]. The main difference lies in partitioning in each dimension separately instead of doing it for the whole state space. Absorbing states are also assigned to each dimension separately instead of having only one for the unsafe set. Moreover, Algorithm 1 stores the transition probabilities efficiently as a BN.

3.4 Probabilistic invariance for the abstract DBN

We extend the use of $\mathbb{P}$ by denoting the probability measure on the set of events defined over a DBN with discrete r.v. $\mathbf{z} = (X_1, X_2, \ldots, X_n)$. Given a discrete set $Z_{\mathfrak{a}} \subset \prod_i \Omega_i$,

the probabilistic invariance problem asks to evaluate the probability $p_N(z_0, Z_a)$ that a finite execution associated with the initial condition $z(0) = z_0$ remains within the set Z_a during the finite time horizon $t = 0, 1, 2, \ldots, N$. Formally,

$$p_N(z_0, Z_a) = \mathbb{P}(z(t) \in Z_a, \text{ for all } t = 0, 1, 2, \ldots, N | z(0) = z_0).$$

This probability can be computed by a discrete analogue of the Bellman backward recursion (see [4] for details).

Theorem 1 *Consider value functions* $V_k^d : \prod_i \Omega_i \to [0, 1]$, $k = 0, 1, 2, \ldots, N$, *computed by the backward recursion*

$$V_k^d(z) = \mathbf{1}_{Z_a}(z) \sum_{\bar{z} \in \prod_i \Omega_i} V_{k+1}^d(\bar{z}) \mathbb{P}(\bar{z}|z) \quad k = 0, 1, 2, \ldots, N-1, \tag{8}$$

and initialized with $V_N^d(z) = \mathbf{1}_{Z_a}(z)$. *Then the solution of the invariance problem is characterized as* $p_N(z_0, Z_a) = V_0^d(z_0)$.

The discrete transition probabilities $\mathbb{P}(\bar{z}|z)$ in Eq. (8) are computed by taking the product of the CPD in $\mathcal{T}$. More specifically, for any $z, \bar{z} \in \prod_i \Omega_i$ of the form $z = (z_1, z_2, \ldots, z_n)$, $\bar{z} = (\bar{z}_1, \bar{z}_2, \ldots, \bar{z}_n)$ we have

$$\mathbb{P}(\bar{z}|z) = \prod_i T_i(\bar{X}_i = \bar{z}_i | Pa(\bar{X}_i) = z).$$

Our algorithm for probabilistic invariance computes $p_N(z_0, Z_a)$ to approximate $p_N(s_0, A)$, for suitable choices of z_0 and Z_a depending on s_0 and A. The natural choice for the initial state is $z_0 = (z_1(0), \ldots, z_n(0))$ with $z_i(0) = \xi_i(\Pi_i(s_0))$. For A, the n-fold Cartesian product of the collection of the partition sets $\{D_{ij}\}$, $i \in \mathbb{N}_n$ generates a cover of A as

$$\begin{aligned} A &\subset \bigcup \{D_{1j}\}_{j=1}^{n_1} \times \{D_{2j}\}_{j=1}^{n_2} \times \ldots \times \{D_{nj}\}_{j=1}^{n_n} \\ &= \bigcup_{\boldsymbol{j}} \left\{ D_{\boldsymbol{j}} | \boldsymbol{j} = (j_1, j_2, \ldots, j_n), D_{\boldsymbol{j}} \doteq D_{1j_1} \times D_{2j_2} \times \ldots \times D_{nj_n} \right\}. \end{aligned}$$

We define the safe set Z_a of the DBN as

$$Z_a = \bigcup_{\boldsymbol{j}} \left\{ (z_{1j_1}, z_{2j_2}, \ldots, z_{nj_n}), \text{ such that } A \cap D_{\boldsymbol{j}} \neq \emptyset \text{ for } \boldsymbol{j} = (j_1, j_2, \ldots, j_n) \right\}, \tag{9}$$

which is a discrete representation of the continuous set $\bar{A} \subset \mathbb{R}^n$

$$\bar{A} = \bigcup_{\boldsymbol{j}} \left\{ D_{\boldsymbol{j}}, \text{ such that } \boldsymbol{j} = (j_1, j_2, \ldots, j_n), A \cap D_{\boldsymbol{j}} \neq \emptyset \right\}. \tag{10}$$

For instance $\bar{A}$ can be a finite union of hypercubes in $\mathbb{R}^n$ if the partition sets D_{ij} are intervals. It is clear that the set $\bar{A}$ is in general different form A.

There are thus two sources of error: first due to replacing A with $\bar{A}y$, and second, due to the abstraction of the dynamics between the discrete outcome obtained by Theorem 1 and the continuous solution that results from (5). In the next section we provide a quantitative bound on the two sources of error.

3.5 Quantification of the error due to abstraction

Let us explicitly write the Bellman recursion (5) of the safety problem over the set $\bar{A}$:

$$W_N(\boldsymbol{s}) = \mathbf{1}_{\bar{A}}(\boldsymbol{s}), \quad W_k(\boldsymbol{s}) = \int_{\bar{A}} W_{k+1}(\bar{\boldsymbol{s}}) t_{\mathfrak{s}}(\bar{\boldsymbol{s}}|\boldsymbol{s}) d\bar{\boldsymbol{s}}, \quad k = 0, 1, 2, \ldots, N-1, \tag{11}$$

which results in $p_N(\boldsymbol{s}_0, \bar{A}) = W_0(\boldsymbol{s}_0)$. Theorem 2 characterizes the error due to replacing the safe set A by $\bar{A}$.

Theorem 2 *Solution of the probabilistic invariance problem with the time horizon N and two safe sets A, $\bar{A}$ satisfies the inequality*

$$|p_N(\boldsymbol{s}_0, A) - p_N(\boldsymbol{s}_0, \bar{A})| \leq MN\mathcal{L}(A \Delta \bar{A}), \quad \forall \boldsymbol{s}_0 \in A \cap \bar{A},$$

where $M \doteq \sup\left\{t_{\mathfrak{s}}(\bar{\boldsymbol{s}}|\boldsymbol{s}) \middle| \boldsymbol{s}, \bar{\boldsymbol{s}} \in A \Delta \bar{A}\right\}$, $\mathcal{L}(B)$ denotes the Lebesgue measure of any set $B \in \mathcal{B}$, and $A \Delta \bar{A} \doteq (A \backslash \bar{A}) \cup (\bar{A} \backslash A)$ is the symmetric difference of the two sets A, $\bar{A}$.

Proof Recall the recursive equations for the probabilistic safety problem over sets A and $\bar{A}$ as in (5) and (11), respectively. Solutions of the safety problems are $p_N(\boldsymbol{s}_0, A) = V_0(\boldsymbol{s}_0)$ and $p_N(\boldsymbol{s}_0, \bar{A}) = W_0(\boldsymbol{s}_0)$. We prove inductively that the inequality $|V_k(\boldsymbol{s}) - W_k(\boldsymbol{s})| \leq M(N-k)\mathcal{L}(\bar{A} \Delta A)$ holds for all $\boldsymbol{s} \in A \cap \bar{A}$. This inequality is true for $k = N$ since $V_N(\boldsymbol{s}) = W_N(\boldsymbol{s}) = 1$ for $\boldsymbol{s} \in A \cap \bar{A}$. For any $k = 0, 1, 2, \ldots, N-1$ and any state $\boldsymbol{s} \in A \cap \bar{A}$ we have

$$\begin{aligned}
|V_k(\boldsymbol{s}) - W_k(\boldsymbol{s})| &\leq \int_{A \cap \bar{A}} |V_{k+1}(\bar{\boldsymbol{s}}) - W_{k+1}(\bar{\boldsymbol{s}})| t_{\mathfrak{s}}(\bar{\boldsymbol{s}}|\boldsymbol{s}) d\bar{\boldsymbol{s}} \\
&\quad + \int_{A \backslash \bar{A}} V_{k+1}(\bar{\boldsymbol{s}}) t_{\mathfrak{s}}(\bar{\boldsymbol{s}}|\boldsymbol{s}) d\bar{\boldsymbol{s}} + \int_{\bar{A} \backslash A} W_{k+1}(\bar{\boldsymbol{s}}) t_{\mathfrak{s}}(\bar{\boldsymbol{s}}|\boldsymbol{s}) d\bar{\boldsymbol{s}} \\
&\leq M(N-k-1)\mathcal{L}(\bar{A} \Delta A) + M\mathcal{L}(\bar{A} \backslash A) + M\mathcal{L}(A \backslash \bar{A}) \\
&= M(N-k)\mathcal{L}(\bar{A} \Delta A).
\end{aligned}$$

The inequality for $k = 0$ proves the upper bound $MN\mathcal{L}(\bar{A} \Delta A)$ on $|p_N(\boldsymbol{s}_0, A) - p_N(\boldsymbol{s}_0, \bar{A})|$. □

The second contribution to the error is related to the discretization of Algorithm 1 which is quantified by posing regularity conditions on the dynamics of the process. The following Lipschitz continuity assumption restricts the generality of the density functions t_k characterizing the dynamics of model $\mathscr{M}_{\mathfrak{s}}$.

Assumption 1 Assume the density functions $t_k(\bar{s}_i|\cdot)$ are Lipschitz continuous with the finite positive Lipschitz constants d_{ij}:

$$|t_j(\bar{s}_j|\boldsymbol{s}) - t_j(\bar{s}_j|\boldsymbol{s}')| \leq d_{ij}|s_i - s_i'|, \tag{12}$$

with $\boldsymbol{s} = [s_1, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_n]$ and $\boldsymbol{s}' = [s_1, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_n]$, for all $s_k, s_k', \bar{s}_k \in D_k$, $k \in \mathbb{N}_n$, and for all $i, j \in \mathbb{N}_n$.

Note that Assumption 1 holds with $d_{ij} = 0$ if and only if $(X_i, \bar{X}_j) \notin \mathcal{E}$ in the DAG of the BN $\mathfrak{B}_{\rightarrow}$. Assumption 1 enables us to assign non-zero weights $w_{ij} = d_{ij}\mathcal{L}(D_j)$ to the arcs $(X_i, \bar{X}_j) \in \mathcal{E}$, for all $i, j \in \mathbb{N}_n$, of the graph. We define the out-weight of the node X_i by $\mathcal{O}_i = \sum_{j=1}^n w_{ij}$ and the in-weight of the node $\bar{X}_j$ by $\mathcal{I}_j = \sum_{i=1}^n w_{ij}$.

Remark 2 The Lipschitz constants d_{ij} in (12) can be obtained as an upper bound on the absolute value of the partial derivative (with respect to s_i) of the density function $t_j(\bar{s}_j|\boldsymbol{s})$. Each constant can be computed using symbolic or numeric differentiation, then performing one single local optimization over the partition of interest. Available software like MATLAB can easily be used to automate such computations. The software tool FAUST^2 [14] already employs computation of Lipschitz constants to perform Markov chain abstraction of stochastic systems. Note that Assumption 1 is a mild restriction on the class of stochastic systems under study and is not bound to linear dynamics. For instance, any non-linear system with additive noise $s_j(t+1) = f_j(\boldsymbol{s}(t)) + \zeta(t)$ in which both $f_j(\cdot)$ and the density function of $\zeta(\cdot)$ are Lipschitz continuous, satisfies Assumption 1. □

Remark 3 Additionally, the above assumption implies Lipschitz continuity of the conditional density functions $t_j(\bar{s}_j|\boldsymbol{s})$. Since trivially $|s_i - s_i'| \leq \|\boldsymbol{s} - \boldsymbol{s}'\|$ for all $i \in \mathbb{N}_n$, we obtain

$$|t_j(\bar{s}_j|\boldsymbol{s}) - t_j(\bar{s}_j|\boldsymbol{s}')| \leq \mathcal{H}_j \|\boldsymbol{s} - \boldsymbol{s}'\| \quad \forall \boldsymbol{s}, \boldsymbol{s}' \in \bar{A}, \bar{s}_j \in D_j,$$

where $\mathcal{H}_j = \sum_{i=1}^n d_{ij}$. The density function $t_{\mathfrak{s}}(\bar{\boldsymbol{s}}|\boldsymbol{s})$ is also Lipschitz continuous if the density functions $t_j(\bar{s}_j|\boldsymbol{s})$ are bounded, but the boundedness assumption is not necessary for the results of this paper to hold. □

Assumption 1 enables us to establish Lipschitz continuity of the value functions W_k in (11). This continuity property is essential in proving an upper bound on the discretization error of Algorithm 1, which we shall present in Corollary 1.

Lemma 1 *Consider the value functions $W_k(\cdot)$, $k = 0, 1, 2, \ldots, N$, employed in Bellman recursion (11) of the safety problem over the set $\bar{A}$. Under Assumption 1, these value functions are Lipschitz continuous*

$$|W_k(\boldsymbol{s}) - W_k(\boldsymbol{s}')| \leq \kappa \|\boldsymbol{s} - \boldsymbol{s}'\|, \quad \forall \boldsymbol{s}, \boldsymbol{s}' \in \bar{A},$$

for all $k = 0, 1, 2, \ldots, N$ with the constant $\kappa = \sum_{j=1}^n \mathcal{I}_j$, where $\mathcal{I}_j$ is the in-weight of the node $\bar{X}_j$ in the DAG of the BN $\mathfrak{B}_{\rightarrow}$.

Proof The inequality holds for $k = N$ since $W_N(\boldsymbol{s}) = W_N(\boldsymbol{s}') = 1$ for any $s, \boldsymbol{s}' \in \bar{A}$. For $k = 0, 1, 2, \ldots, N-1$ and any $\boldsymbol{s}, \boldsymbol{s}' \in \bar{A}$ we have

$$\begin{aligned} |W_k(\boldsymbol{s}) - W_k(\boldsymbol{s}')| &\leq \int_{\bar{A}} W_{k+1}(\bar{\boldsymbol{s}}) |t_{\mathfrak{s}}(\bar{\boldsymbol{s}}|\boldsymbol{s}) - t_{\mathfrak{s}}(\bar{\boldsymbol{s}}|\boldsymbol{s}')| d\bar{\boldsymbol{s}} \\ &\leq \int_{\bar{A}} |t_{\mathfrak{s}}(\bar{\boldsymbol{s}}|\boldsymbol{s}) - t_{\mathfrak{s}}(\bar{\boldsymbol{s}}|\boldsymbol{s}')| d\bar{\boldsymbol{s}} \end{aligned}$$

Next, we employ a telescopic sum for the multiplicative structure of the density functions in the integrand on the right-hand side, to obtain:

$$\begin{aligned} |W_k(\boldsymbol{s}) - W_k(\boldsymbol{s}')| &\leq \int_{\bar{A}} \left| \prod_{i=1}^n t_i(\bar{s}_i|\boldsymbol{s}) - \prod_{i=1}^n t_i(\bar{s}_i|\boldsymbol{s}') \right| d\bar{\boldsymbol{s}} \\ &= \int_{\bar{A}} \left| \sum_{j=1}^n \left[\prod_{i=1}^{j-1} t_i(\bar{s}_i|\boldsymbol{s}') \prod_{i=j}^n t_i(\bar{s}_i|\boldsymbol{s}) - \prod_{i=1}^{j} t_i(\bar{s}_i|\boldsymbol{s}') \prod_{i=j+1}^n t_i(\bar{s}_i|\boldsymbol{s}) \right] \right| d\bar{\boldsymbol{s}} \\ &\leq \sum_{j=1}^n \int_{\bar{A}} \left[\prod_{i=1}^{j-1} t_i(\bar{s}_i|\boldsymbol{s}') \prod_{i=j+1}^n t_i(\bar{s}_i|\boldsymbol{s}) \left| t_j(\bar{s}_j|\boldsymbol{s}) - t_j(\bar{s}_j|\boldsymbol{s}') \right| \right] d\bar{\boldsymbol{s}} \end{aligned}$$

$$\leq \sum_{j=1}^{n} \int_{D_j} \left| t_j(\bar{s}_j|\boldsymbol{s}) - t_j(\bar{s}_j|\boldsymbol{s}') \right| d\bar{s}_j$$

$$\leq \sum_{j=1}^{n} \mathcal{H}_j \|\boldsymbol{s} - \boldsymbol{s}'\| \mathcal{L}(D_j) = \|\boldsymbol{s} - \boldsymbol{s}'\| \sum_{j=1}^{n} \mathcal{H}_j \mathcal{L}(D_j) = \|\boldsymbol{s} - \boldsymbol{s}'\| \sum_{j=1}^{n} \mathcal{I}_j. \tag{13}$$

□

Corollary 1 *The following inequality holds under Assumption 1:*

$$|p_N(\boldsymbol{s}_0, A) - p_N(\boldsymbol{z}_0, Z_{\mathfrak{a}})| \leq MN\mathcal{L}(A \Delta \bar{A}) + N\kappa\delta \quad \forall \boldsymbol{s}_0 \in A,$$

where $p_N(\boldsymbol{z}_0, Z_{\mathfrak{a}})$ is the invariance probability for the DBN obtained by Algorithm 1. The initial state of the DBN is $\boldsymbol{z}_0 = (z_1(0), \ldots, z_n(0))$ with $z_i(0) = \xi_i(\Pi_i(\boldsymbol{s}_0))$. The set $Z_{\mathfrak{a}}$ and the constant M are defined in (9) and Theorem 2, respectively. The diameter of the partition of Algorithm 1 is defined and used as

$$\delta = \sup\{\|\boldsymbol{s} - \boldsymbol{s}'\|, \forall \boldsymbol{s}, \boldsymbol{s}' \in D_{\boldsymbol{j}}, \forall \boldsymbol{j} \;\; D_{\boldsymbol{j}} \subset \bar{A}\}.$$

Proof Construction of the set $\bar{A}$ in (10) implies that $A \subseteq \bar{A}$. We use triangle inequality and utilize the bound established in Theorem 2 to get, for all $\boldsymbol{s}_0 \in A$,

$$\begin{aligned} |p_N(\boldsymbol{s}_0, A) - p_N(\boldsymbol{z}_0, Z_{\mathfrak{a}})| &\leq |p_N(\boldsymbol{s}_0, A) - p_N(\boldsymbol{s}_0, \bar{A})| + |p_N(\boldsymbol{s}_0, \bar{A}) - p_N(\boldsymbol{z}_0, Z_{\mathfrak{a}})| \\ &\leq MN\mathcal{L}(A \Delta \bar{A}) + |W_0(\boldsymbol{s}_0) - V_0^d(\boldsymbol{z}_0)|, \end{aligned}$$

where V_0^d and W_0 are defined respectively in (8) and (11). DBN constructed in Sect. 3.3 is in fact a finite-state Markov chain with a specific structure. Combining this with the Lipschitz continuity property of W_0 proved in Lemma 1 enable us to utilize the bound provided in [2]: the error caused by the state-space discretization is upper-bounded by multiplication of three terms, which are Lipschitz constant of the value functions κ, horizon of the invariance specification N, and the diameter δ of the partition selected for the set $\bar{A}$. Then we have $|W_0(\boldsymbol{s}_0) - V_0^d(\boldsymbol{z}_0)| \leq N\kappa\delta$, which completes the proof. □

The second error term in Corollary 1 is a linear function of the partition diameter δ, which depends on all partition sets along different dimensions. We are interested in proving a dimension-dependent error bound in order to parallelize the whole abstraction procedure along different dimensions. The next theorem gives this dimension-dependent error bound.

Theorem 3 *The following inequality holds under Assumption 1:*

$$|p_N(\boldsymbol{s}_0, A) - p_N(\boldsymbol{z}_0, Z_{\mathfrak{a}})| \leq MN\mathcal{L}(A \Delta \bar{A}) + N \sum_{i=1}^{n} \mathcal{O}_i \delta_i \quad \forall \boldsymbol{s}_0 \in A, \tag{14}$$

with the constants defined in Corollary 1. $\mathcal{O}_i$ is the out-weight of the node X_i in the DAG of the BN $\mathfrak{B}_{\rightarrow}$. The quantity δ_i is the maximum diameter of the partition sets along the ith dimension

$$\delta_i = \sup\{|s_i - s_i'|, \forall s_i, s_i' \in D_{ij}, \forall j \in \mathbb{N}_{n_i}\}.$$

Proof The proof follows the same lines as those of Lemma 1. We refine the inequality (13) to obtain an upper bound for $|W_k(\boldsymbol{s}) - W_k(\boldsymbol{s}')|$ localized to partition sets. Namely, for any $\boldsymbol{s}, \boldsymbol{s}' \in D_{\boldsymbol{j}}$,

$$|W_k(\boldsymbol{s}) - W_k(\boldsymbol{s}')| \leq \sum_{j=1}^{n} \int_{D_j} \left| t_j(\bar{s}_j|\boldsymbol{s}) - t_j(\bar{s}_j|\boldsymbol{s}') \right| d\bar{s}_j$$

$$\leq \sum_{j=1}^{n} \int_{D_j} \sum_{i=1}^{n} d_{ij}|s_i - s_i'| d\bar{s}_j \leq \sum_{i,j=1}^{n} d_{ij}\delta_i \mathcal{L}(D_j) = \sum_{i=1}^{n} \mathcal{O}_i \delta_i.$$

Next, we utilize the results of [10], which give an upper bound on the partitioning error based on the above local computation. This implies $|W_0(\boldsymbol{s}_0) - V_0^d(\boldsymbol{z}_0)| \leq N \sum_{i=1}^{n} \mathcal{O}_i \delta_i$. The rest of the proof is exactly the same as that of Corollary 1. □

For a given error threshold ϵ, we can select the set $\bar{A}$ and consequently the diameters δ_i such that $MN\mathcal{L}(A \Delta \bar{A}) + N \sum_{i=1}^{n} \mathcal{O}_i \delta_i \leq \epsilon$. Therefore, generation of the abstract DBN, namely selection of the partition sets $\{D_{ij},\ j \in \mathbb{N}_i\}$ (according to the diameter δ_i) and computation of the CPD, can be implemented in parallel. For a given ϵ and set $\bar{A}$, the cardinality of the state space $\Omega_i, i \in \mathbb{N}_n$, of the discrete random variable X_i and thus the size of the CPD T_i, grow linearly as a function of the horizon of the specification N.

Notice that the constant M defined in Theorem 2 and used in (14) depends on $\bar{A}$ but can be replaced by

$$M_C = \sup \left\{ t_{\bar{\boldsymbol{s}}}(\bar{\boldsymbol{s}}|\boldsymbol{s}) \middle| \boldsymbol{s}, \bar{\boldsymbol{s}} \in C \right\}, \tag{15}$$

where C is any set that contains $A \Delta \bar{A}$. In order to tune the error in (14), one method will be selecting the set C as a box containing the safe set A, computing the constant M_C as in (15), and then choosing $\bar{A}$ such that $A \subseteq \bar{A} \subseteq C$ with a suitable $\mathcal{L}(\bar{A} \Delta A)$. Subsequently, the partition diameters δ_i are selected for this set $\bar{A}$ to guarantee the error threshold ϵ.

4 Efficient model checking of the finite-state DBN

Existing numerical methods for model checking DBNs with discrete r.v. transform the DBN into an explicit matrix representation [17,23,26], which defeats the purpose of a compact representation. Instead, we show that the multiplicative structure of the transition probability matrix can be incorporated in the computation which makes the construction of $\mathbb{P}(\bar{\boldsymbol{z}}|\boldsymbol{z})$ dispensable. For this purpose we employ *factor graphs* and the *sum-product algorithm* [21] originally developed for marginalizing functions and applied to belief propagation in Bayesian networks. Suppose that a *global* function is given as a product of *local* functions, and that each local function depends on a subset of the variables of the global map. In its most general form, the sum-product algorithm acts on factor graphs in order to marginalize the global function, i.e., taking summation respect to a subset of variables, exploiting its product structure [21]. In our problem, we restrict the summation domain of the Bellman recursion (8) to $\prod_i Z_i$ because the value functions are simply equal to zero in the complement of this set. The summand in (8) has the multiplicative structure

$$g(\boldsymbol{z}, \bar{\boldsymbol{z}}) \doteq \mathbf{1}_{Z_a}(\boldsymbol{z}) V_{k+1}^d(\bar{\boldsymbol{z}}) \prod_i T_i(\bar{X}_i = \bar{z}_i | Pa(\bar{X}_i) = \boldsymbol{z}), \quad V_k^d(\boldsymbol{z}) = \sum_{\bar{\boldsymbol{z}} \in \prod_i Z_i} g(\boldsymbol{z}, \bar{\boldsymbol{z}}). \tag{16}$$

The function $g(\boldsymbol{z}, \bar{\boldsymbol{z}})$ depends on variables $\{z_i, \bar{z}_i,\ i \in \mathbb{N}_n\}$. The factor graph of $g(\boldsymbol{z}, \bar{\boldsymbol{z}})$ has $2n$ *variable nodes*, one for each variable and $(n + 2)$ *function nodes* for local functions $\mathbf{1}_{Z_a}, V_{k+1}^d, T_i$. An arc connects a variable node to a function node if and only if the variable is an argument of the local function. The factor graph of Example 4 for $n = 4$ is presented in Fig. 3—factor graphs of general functions $g(\boldsymbol{z}, \bar{\boldsymbol{z}})$ in (16) are similar to that in Fig. 3, the

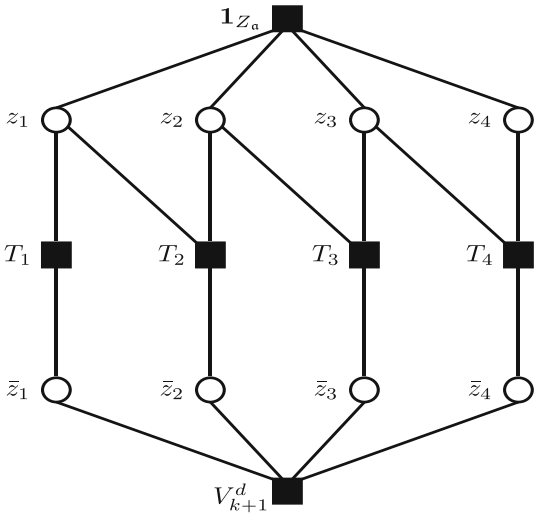

Fig. 3 Factor graph of the linear stochastic system (7) for $n = 4$

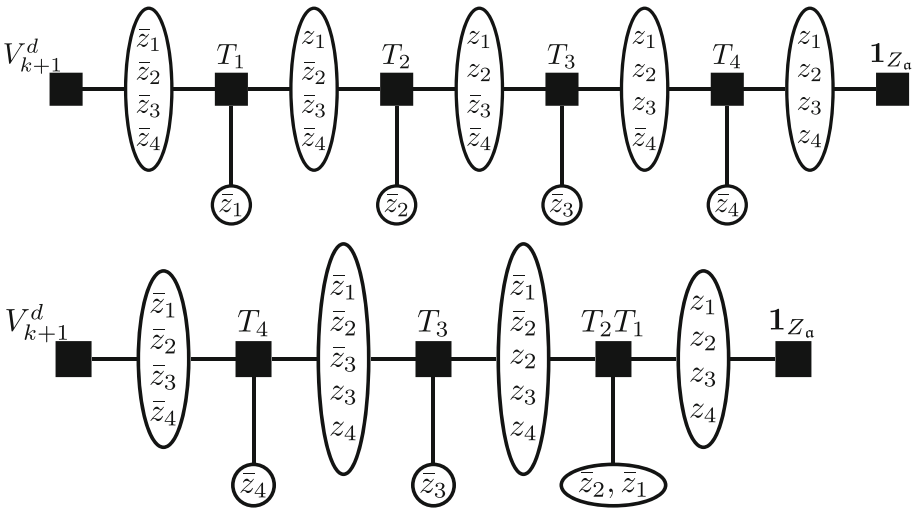

Fig. 4 Spanning tree of the linear stochastic system in (7) for $n = 4$ and two orderings $(\bar{z}_4, \bar{z}_3, \bar{z}_2, \bar{z}_1)$ (*top plot*) and $(\bar{z}_1, \bar{z}_2, \bar{z}_3, \bar{z}_4)$ (*bottom plot*)

only part needing to be modified being the set of arcs connecting variable nodes $\{z_i,\ i \in \mathbb{N}_n\}$ and function nodes $\{T_i,\ i \in \mathbb{N}_n\}$. This part of the graph can be obtained from the DAG of $\mathfrak{B}_{\rightarrow}$ of the DBN.

The factor graph of a function $g(z, \bar{z})$ contains loops for $n \geq 2$ and must be transformed to a *spanning tree* using clustering and stretching transformations [21]. For this purpose the order of clustering function nodes $\{T_i, i \in \mathbb{N}_n\}$ and that of stretching variable nodes $\{z_i,\ i \in \mathbb{N}_n\}$ needs to be chosen. Figure 4 presents the spanning trees of the stochastic system in (7) for two such orderings. The variable nodes at the bottom of each spanning tree specify the order of the summation, whereas the function nodes considered from the left to the right indicate the order of multiplication of the local functions. The rest of the variable nodes show the arguments of the intermediate functions, which reflects the required memory for storing

Algorithm 2 Greedy algorithm for obtaining the order of stretching variables and clustering functions in the factor graph

Require: Factor graph of the summand in Bellman recursion
1: Initialize the sets $\mathcal{U}_1 = \{z_i,\ i \in \mathbb{N}_n\}, \mathcal{U}_2 = \{\bar{z}_i,\ i \in \mathbb{N}_n\}, \mathcal{U}_3 = \{T_i,\ i \in \mathbb{N}_n\}, e_{\mathfrak{f}} = \kappa_{\mathfrak{f}} = \emptyset$
2: **while** $\mathcal{U}_1 \neq \emptyset$ **do**
3: For any node $u \in \mathcal{U}_3$ compute $Pa_{\mathfrak{f}}(u)$ (resp. $Ch_{\mathfrak{f}}(u)$) as the elements of $\mathcal{U}_1$ (resp. $\mathcal{U}_2$) connected to u by an arc in the factor graph
4: Define the equivalence relation R on $\mathcal{U}_3$ as $uR\bar{u}$ iff $Pa_{\mathfrak{f}}(u) = Pa_{\mathfrak{f}}(\bar{u})$
5: Replace the set $\mathcal{U}_3$ with the set of equivalence classes induced by R.
6: Combine all the variable nodes of $Ch_{\mathfrak{f}}(u)$ connected to one class
7: Select $u \in \mathcal{U}_3$ with the minimum cardinality of $Pa_{\mathfrak{f}}(u)$ and put $e_{\mathfrak{f}} = (u, e_{\mathfrak{f}}), \kappa_{\mathfrak{f}} = (Ch_{\mathfrak{f}}(u), \kappa_{\mathfrak{f}})$
8: Update the sets $\mathcal{U}_1 = \mathcal{U}_1 \backslash Pa_{\mathfrak{f}}(u), \mathcal{U}_2 = \mathcal{U}_2 \cup Pa_{\mathfrak{f}}(u) \backslash Ch_{\mathfrak{f}}(u), \mathcal{U}_3 = \mathcal{U}_3 \backslash \{u\}$, and eliminate all the arcs connected to u
9: **end while**
Ensure: The order of variables $\kappa_{\mathfrak{f}}$ and functions $e_{\mathfrak{f}}$

such functions. The computational complexity of the solution carried out on the spanning tree clearly depends on this ordering.

Algorithm 2 presents a greedy procedure that operates on the factor graph and provides an ordering of the variables and of the functions to reduce the overall memory usage. This algorithm iteratively combines the function nodes and selects the next variable node, over which the summation is carried out. Step 1 initializes the algorithm by distinguishing three sets of nodes: $\mathcal{U}_1 = \{z_1, z_2, \ldots, z_n\}$ and $\mathcal{U}_2 = \{\bar{z}_1, \bar{z}_2, \ldots, \bar{z}_n\}$ contain the variable nodes and $\mathcal{U}_3 = \{T_1, T_2, \ldots, T_n\}$ includes function nodes. The *sequences* $e_{\mathfrak{f}}$ and $\kappa_{\mathfrak{f}}$ are initially empty and will contain the function and variables for performing product and sum in the sum-product algorithm. These sequences are built progressively during the while loop of the algorithm.

In each iteration of the while loop we compute the set of nodes from $\mathcal{U}_1$ and $\mathcal{U}_2$ connected to the elements of $\mathcal{U}_3$ through functions $Pa_{\mathfrak{f}}$ and $Ch_{\mathfrak{f}}$, respectively. Steps 4–6 modify the graph to combine the nodes in $\mathcal{U}_3$ that are connected to the same set of nodes in $\mathcal{U}_1$ since these function nodes have the same conditional variables and their memory usage is exactly the same. Step 7 selects the next function and variable nodes for performing product and sum in the sum-product algorithm such that the required memory is minimal among the possible selections. Finally, step 8 updates the sets after such selection.

Note that Algorithm 2 is applied to the factor graph of the system which has only $(3n+2)$ nodes. In contrast, the memory usage of the DBN model checking is a polynomial function of the number of partition sets which is in general much larger than $(3n + 2)$ for practical accuracies. Thus the overhead related to Algorithm 2 is definitely worth when viewed from the perspective of the attained memory savings. Since Algorithm 2 computes the ordering progressively, its outcome depends on the structure of the factor graph and is sub-optimal in general. The output of this algorithm implemented on the factor graph of Example 4 is the orderings $\kappa_{\mathfrak{f}} = (\bar{z}_4, \bar{z}_3, \bar{z}_2, \bar{z}_1)$ and $e_{\mathfrak{f}} = (T_4, T_3, T_2, T_1)$, started from the outermost sum, which is related to the spanning tree on top of Fig. 4.

5 Comparison with the state of the art

In this section we compare our approach with the state-of-the-art abstraction procedure presented in [2] (referred to as AKLP in the following), which does not exploit the structure

of the dynamics. The AKLP algorithm approximates the concrete model with a finite-state Markov chain by uniformly gridding the safe set. As in our work, the error bound of the AKLP procedure depends on the global Lipschitz constant of the density function of the model, however it does not exploit its structure as proposed in this work. We compare the two procedures on (1) error bounds and (2) computational resources.

Consider the stochastic linear dynamical model in (7), where $\Phi = [a_{ij}]_{i,j}$ is an arbitrary matrix. The Lipschitz constants d_{ij} in Assumption 1 can be computed as $d_{ij} = |a_{ji}|/\sigma_j^2\sqrt{2\pi e}$, where e is Euler's constant. From Theorem 3, we get the following error bound:

$$e_{\mathrm{DBN}} \doteq MN\mathcal{L}(A\Delta\bar{A}) + \frac{N}{\sqrt{2\pi e}} \sum_{i,j=1}^{n} \frac{|a_{ji}|}{\sigma_j^2} \mathcal{L}(D_j)\delta_i.$$

On the other hand, the error bound for AKLP is

$$e_{\mathrm{AKLP}} = MN\mathcal{L}(A\Delta\bar{A}) + \frac{Ne^{-1/2}}{(\sqrt{2\pi})^n\sigma_1\sigma_2\cdots\sigma_n} \|\Sigma^{-1/2}\Phi\|_2\delta\mathcal{L}(A).$$

In order to meaningfully compare the two error bounds, select set $A = [-\alpha, \alpha]^n$ and $\sigma_i = \sigma, i \in \mathbb{N}_n$, and consider hypercubes as partition sets. The two error terms then become

$$e_{\mathrm{DBN}} = \varsigma n\eta \left(\frac{\|\Phi\|_1}{n\sqrt{n}}\right), \quad e_{\mathrm{AKLP}} = \varsigma\eta^n\|\Phi\|_2, \quad \eta = \frac{2\alpha}{\sigma\sqrt{2\pi}}, \quad \varsigma = \frac{N\delta}{\sigma\sqrt{e}},$$

where $\|\Phi\|_1$ and $\|\Phi\|_2$ are the entry-wise one-norm and the induced two-norm of matrix Φ, respectively. The error e_{AKLP} depends exponentially on the dimension n as η^n, whereas we have reduced this term to a linear one $(n\eta)$ in our proposed new approach resulting in error e_{DBN}. Note that $\eta \leq 1$ means that the standard deviation of the process noise is larger than the selected safe set: in this case the value functions (which characterize the probabilistic invariance problem) uniformly converge to zero with rate η^n; clearly the case of $\eta > 1$ is more interesting. On the other hand for any matrix Φ we have $\frac{\|\Phi\|_1}{n\sqrt{n}} \leq \|\Phi\|_2$. This second term indicates how sparsity is reflected in the error computation. Denote by r the degree of connectivity of the DAG of $\mathfrak{B}_{\rightarrow}$ for this linear system, which is the maximum number of non-zero elements in rows of matrix Φ. We adapt the following inequalities from [20] for the norms of matrix Φ (refer to the Appendix for a formal proof):

$$\|\Phi\|_2 \leq \sqrt{nr} \max_{i,j} |a_{ij}|, \qquad \frac{\|\Phi\|_1}{n\sqrt{n}} \leq \frac{r}{\sqrt{n}} \max_{i,j} |a_{ij}|, \tag{17}$$

which shows that for a fixed dimension n, sparse dynamics, compared to fully connected dynamics, results in better error bounds in the new approach.

In order to compare computational resources, consider the numerical values $N = 10$, $\alpha = 1$, $\sigma = 0.2$, and the error threshold $\epsilon = 0.2$ for the lower bidiagonal matrix Φ with all the non-zero entries set to one. Table 1 compares the number of required partition sets (or bins) per dimension, the number of marginals, and the required number of (addition and multiplication) operations for the verification step, for models of different dimensions (number of continuous variables n). The numerical values in Table 1 confirm that for a given upper bound on the error ϵ, the number of bins per dimension and the required marginals grow exponentially in dimension for AKLP and polynomially for our DBN-based approach. For instance, to ensure the error is at most ϵ for the model of dimension $n = 4$, the cardinality of the partition of each dimension for the uniform gridding and for the structured approach is 2.9×10^5 and 8.5×10^3, respectively. Then, AKLP requires storing 4.8×10^{43} entries (which

Table 1 Comparison of the AKLP and the DBN-based algorithms, over the stochastic linear dynamical model (7)

Dimension n		1	2	3	4	5	6	7	8
# Bins/dim	AKLP	1.2×10^3	1.1×10^4	6.0×10^4	2.9×10^5	1.3×10^6	5.8×10^6	2.5×10^7	1.1×10^8
	DBN	1.2×10^3	3.6×10^3	6.0×10^3	8.5×10^3	1.1×10^4	1.3×10^4	1.6×10^4	1.8×10^4
# Marginals	AKLP	1.5×10^6	1.5×10^{16}	4.8×10^{28}	4.8×10^{43}	1.5×10^{61}	1.5×10^{81}	4.3×10^{103}	3.5×10^{128}
	DBN	1.5×10^6	4.8×10^{10}	4.4×10^{11}	1.8×10^{12}	5.2×10^{12}	1.2×10^{13}	2.3×10^{13}	4.2×10^{13}
# Operations	AKLP	2.9×10^7	3.1×10^{17}	1.0×10^{30}	1.1×10^{45}	3.7×10^{62}	3.7×10^{82}	1.1×10^{105}	9.5×10^{129}
	DBN	2.9×10^7	1.9×10^{12}	8.0×10^{16}	3.5×10^{21}	1.7×10^{26}	8.9×10^{30}	5.2×10^{35}	3.4×10^{40}

The number of partition sets (or bins) per dimension, the number of marginals, and the total required number of (addition and multiplication) operations for the verification step, are compared for models of different dimensions (number of continuous variables n)

is infeasible!), whereas the DBN approach requires 1.8×10^{12} entries ($\sim$ 8GB). The number of operations required for computation of the safety probability are 1.1×10^{45} and 3.5×10^{21}, respectively. This shows a substantial reduction in memory usage and computational time effort: with given memory and computational resources, the DBN-based approach in compare with AKLP promises to handle systems with dimension that is at least twice as large.

Statistical model checking (SMC) [24] is an alternative approach to analyse general probabilistic systems against temporal specifications. Our approach is distinct from SMC in the type of guarantees we provide on the numerical outcomes: namely, our approach provides absolute guarantees for satisfaction of the safety specification as in Corollary 1, whereas SMC provides probabilistic guarantees (i.e., with a given confidence). Moreover, we can compute safety probabilities for any initial state of the process belonging to a continuous domain, but the SMC approach can handle only a finite set of initial states and its computational complexity is linear in the cardinality of the initial set. Therefore SMC by itself cannot handle continuous domains of initial state as we do in this article. To address this, one option would be to partition the set of initial states, verify the process for representative points of the partition sets, and then perform a sensitivity analysis to judge satisfaction of the specification for non-evaluated initial states. Such a sensitivity analysis can be seen as a special case of our approach (i.e., using the Lipschitz continuity of the density function to prove that the property is a smooth function of the initial state). Finally, our approach can be as well extended to models with non-determinism, which is a feature knowingly difficult for existing SMC algorithms and tools.

6 Numerical case study

In this section we present a model for a metabolic network [1] based on a stochastic process, and compute the invariance probability over the model. A metabolic reaction network consists of a set of $\mathfrak{c}$ metabolites and a related set of $\mathfrak{b}$ fluxes between the metabolites in pool $\boldsymbol{c}$. The concentrations of the metabolites are represented with a vector $\boldsymbol{c} \in (\mathbb{R}_{\geq 0})^{\mathfrak{c}}$, and the set of fluxes is denoted by a vector $\boldsymbol{v} \in \mathbb{R}^{\mathfrak{b}}$. The metabolic network considered in this section is adapted from [1] and displayed in Fig. 5.

The material fluxes in $\boldsymbol{v}$ depend on enzymatic reaction mechanisms (for instance, Michaelis-Menten kinetics), substrate concentrations and allosteric effectors (vector $\boldsymbol{c}$), and parameters of the mechanisms ($\boldsymbol{\alpha}$, encompassing for instance affinity constants, maximal conversion rates, etc). The rates of change for concentrations in $\boldsymbol{c}$ are described by balancing the in- and out-fluxes for each metabolite pool. These balances can be expresses via a stoichiometric matrix $N_{\mathfrak{r}} \in \mathbb{Z}^{\mathfrak{c} \times \mathfrak{b}}$, which relates the number of balanced metabolites to the reactions present in the network, and via the flux functions in vector $\boldsymbol{v}$. These fluxes depend on the metabolite concentrations $\boldsymbol{c}$, as well as on the physio-chemical parameters $\boldsymbol{\alpha}$ (e.g. kinetic parameters), and on additional parameters $\boldsymbol{\beta}$ encompassing operational variables (e.g. substrate feed to the reactor, dilution rates, and other experimental settings), as follows:

$$d\boldsymbol{c} = N_{\mathfrak{r}} \boldsymbol{v}(\boldsymbol{c}, \boldsymbol{\alpha}, \boldsymbol{\beta}) dt + d\boldsymbol{w}_t,$$

where $\{\boldsymbol{w}_t, t \in \mathbb{R}_{\geq 0}\}$ is a Wiener process that additively captures uncertainties in the parameters and in the unmodeled dynamics of the metabolic network. A discrete-time dynamical model can be obtained by time sampling via the known Euler-Maruyama scheme, which yields

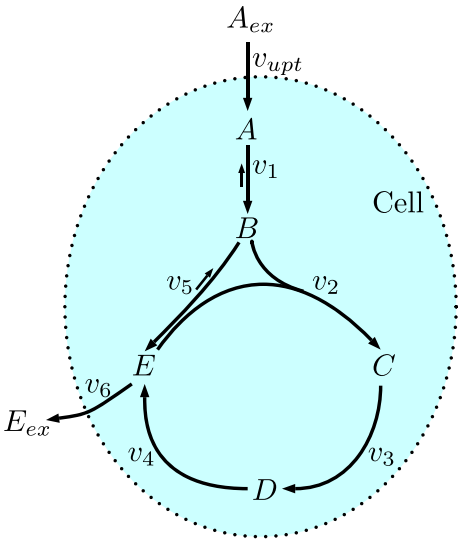

Fig. 5 Metabolic network considered for case study presented in Sect. 6. The network presents two extracellular metabolites (A_{ex} and E_{ex}), and five intracellular ones (A to E). *Arrows* are labeled with metabolic fluxes, affecting the metabolites concentrations dynamics as per (18)

$$\boldsymbol{c}(t+1) = \boldsymbol{c}(t) + N_{\mathfrak{r}}\boldsymbol{v}(\boldsymbol{c}(t), \boldsymbol{\alpha}, \boldsymbol{\beta})\tau + \boldsymbol{\zeta}(t), \tag{18}$$

where τ is the sample time and $\{\boldsymbol{\zeta}(t),\ t \in \mathbb{N}\}$ is an iid Gaussian random sequence.

The state vector denoting the concentrations of the metabolites in (18) is $\boldsymbol{c} = [c_A, c_B, c_C, c_D, c_E]^T \in \mathbb{R}^5$. The stoichiometric matrix $N_{\mathfrak{r}}$ can be written as

$$N_{\mathfrak{r}} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix},$$

and the fluxes vector $\boldsymbol{v} = [v_{upt}, v_1, v_2, v_3, v_4, v_5, v_6]^T$, where v_{upt} is assumed to be a constant input flux. The structure and parameters of the kinetic equations are reported in Tables 2 and 3, respectively.

The one-step conditional density function of the network is a multivariate Gaussian $t_{\mathfrak{s}}(\bar{\boldsymbol{c}}|\boldsymbol{c}) \sim \mathcal{N}(\boldsymbol{m}(\boldsymbol{c}), \Sigma)$, with a mean $\boldsymbol{m}(\boldsymbol{c})$ that depends on the state vector $\boldsymbol{c}$ as follows:

$$\boldsymbol{m}(\boldsymbol{c}) = \begin{bmatrix} m_1(c_A, c_B) \\ m_2(c_A, c_B, c_E) \\ m_3(c_B, c_C, c_E) \\ m_4(c_C, c_D) \\ m_5(c_B, c_D, c_E) \end{bmatrix} = \begin{bmatrix} c_A + \tau\left[v_{upt} - v_1(c_A, c_B)\right] \\ c_B + \tau\left[v_1(c_A, c_B) - v_2(c_B, c_E) - v_5(c_B, c_E)\right] \\ c_C + \tau\left[v_2(c_B, c_E) - v_3(c_C)\right] \\ c_D + \tau\left[v_3(c_C) - v_4(c_D)\right] \\ c_E + \tau\left[v_4(c_D) + v_5(c_B, c_E) - v_6(c_E)\right] \end{bmatrix},$$

and with a covariance matrix Σ of $\{\boldsymbol{\zeta}(t)\}$. The two-layered BN $\mathfrak{B}_{\rightarrow}$ associated with the metabolic network (18) is presented in Fig. 6.

We assume the noises affecting reaction equations in (18) are independent [15], which makes the covariance matrix diagonal $\Sigma = diag([\sigma_1^2, \sigma_2^2, \ldots, \sigma_5^2])$. We use Lemma 2 in the appendix to compute weights w_{ij} associated with the DAG of $\mathfrak{B}_{\rightarrow}$. These weights can be written as $w_{ij} = 2h_{ij}/(\sigma_i\sqrt{2\pi})$, where h_{ij} is the Lipschitz constant of the mean $m_i(\boldsymbol{c})$ respect to the j^{th} element of $\boldsymbol{c}$. We consider the safe set $A = [0, 1]^5$, time step $\tau = 0.05$, time horizon $N = 8$, input flux $v_{upt} = 0.8$, and standard deviations $\sigma_i = 0.2$. The number of bins

Table 2 Kinetic equations in the metabolic network of the case study in Sect. 6

Flux	Mechanism	Kinetic equation
v_1	Reversible Michaelis–Menten	$\frac{v_{max}\left(c_A - \frac{c_B}{K_{eq}}\right)}{K_{mA}\left(1 + \frac{c_B}{K_{mP}}\right) + c_A}$
v_2	2 Substrate Hill-Kinetic	$\frac{v_{max} c_B^{hA} c_E^{hB}}{\left(K_{mA} + c_B^{hA}\right)\left(K_{mB} + c_E^{hB}\right)}$
v_3	Michaelis–Menten	$\frac{v_{max} c_C}{K_{mA} + c_C}$
v_4	Michaelis–Menten	$\frac{v_{max} c_D}{K_{mA} + c_D}$
v_5	Reversible Michaelis–Menten	$\frac{v_{max}\left(c_B - \frac{c_E}{K_{eq}}\right)}{K_{mA}\left(1 + \frac{c_E}{K_{mP}}\right) + c_B}$
v_6	Michaelis–Menten	$\frac{v_{max} c_E}{K_{mA} + c_E}$

Table 3 Parameter values used for the metabolic network of the case study in Sect. 6 (all parameters v_{max} have unit $[\mu mol/(gCDW.s)]$, K_{mA} and K_{mB} in the kinetic equation of v_2 respectively have units $[(\mu mol/gCDW)^{hA}]$ and $[(\mu mol/gCDW)^{hB}]$, all other K_{mA}, K_{mP} have the metabolite concentration unit $[\mu mol/gCDW]$)

Reaction	Parameter	Value	Reaction	Parameter	Value
v_1	v_{max}	3	v_2	v_{max}	2.5
	K_{eq}	3		K_{mA}	0.25
	K_{mA}	0.1		hA	2
	K_{mP}	3		K_{mB}	2
				hB	3
v_3	v_{max}	2	v_4	v_{max}	3
	K_{mA}	2		K_{mA}	2
v_5	v_{max}	2	v_6	v_{max}	2
	K_{eq}	4		K_{mA}	3
	K_{mA}	1			
	K_{mP}	1			

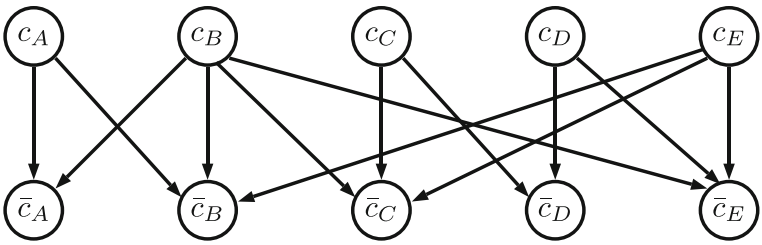

Fig. 6 Two-layered BN $\mathfrak{B}_{\rightarrow}$ associated with the metabolic network of Fig. 5, with dynamics in (18)

per dimension $[55, 13, 8, 8, 9] \times 10^2$ is required to guarantee the error threshold $\epsilon = 0.2$. Solution of the invariance problem is presented in Fig. 7. Each plot represents the solution as a function of two initial state variables where the other three initial states are zero. These

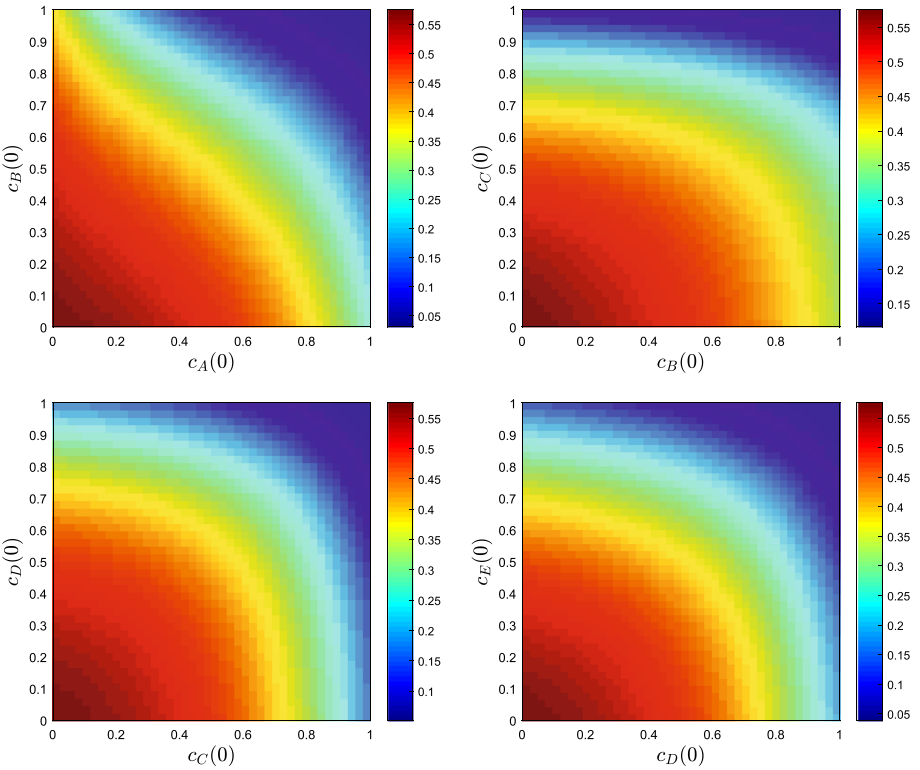

Fig. 7 Solution of the probabilistic invariance problem for the case study of Sect. 6, as a function of initial states of the process. Each *plot* represents the solution as a function of two initial metabolite concentrations (with units $[\mu mol/gCDW]$), where the other three initial concentrations have been taken to be equal to zero

simulation results indicate that the concentrations of all metabolites remain within the interval $[0, 1]$ during the time horizon $N = 8$ with high probability for initial concentrations close to zero. The probability decreases for initial concentrations close to one. In other words, the noise term in Eq. (18) forces the concentrations to jump outside of the interval with a higher chance for initial concentrations close to the upper limit of the interval.

7 Conclusions and future directions

While we have focused on probabilistic invariance, our abstraction approach can be extended to more general properties expressed within the bounded-horizon fragment of PCTL [28] or to bounded-horizon linear temporal properties [3,30], since the model checking problem for these logics reduce to computations of value functions similar to the Bellman recursion scheme. Our focus in this paper has been the foundations of DBN-based abstraction for general Markov processes: factored representations, error bounds, and algorithms. We are currently implementing these algorithms in the FAUST^2 tool [14], and scaling the algorithms using dimension-dependent adaptive gridding [10] as well as implementations of the sum-product algorithm on top of data structures such as algebraic decision diagrams (as in probabilistic model checkers [22]).

Acknowledgements Funding was provided by the European Commission (IAPP Project AMBI 324432) and Oxford University Press (John Fell OUP Research Fund).

8 Appendix A

Proof of Eq. (17). We utilize an inequality from [20], which provides bounds for the maximum singular value of a matrix based on its sparsity pattern. In particular, for any $\Phi = [a_{ij}]_{i,j} \in \mathbb{R}^{n\times n}$,

$$\|\Phi\|_2 \leq \max_{i,j:a_{ij}\neq 0} [r_i(\Phi)c_j(\Phi)]^{1/2},$$

where $r_i(\Phi) = \sum_{j=1}^n |a_{ij}|$ and $c_j(\Phi) = \sum_{i=1}^n |a_{ij}|$. The degree of connectivity of the DAG of $\mathfrak{B}_{\rightarrow}$ for the linear system under study in this section is exactly the maximum number of nonzero entries in the rows of the matrix Φ, which results in $r_i(\Phi) \leq r \max_{i,j} |a_{ij}|$ and $c_j(\Phi) \leq n \max_{i,j} |a_{ij}|$. Using the above inequality we obtain the first inequality in (17), namely

$$\|\Phi\|_2 \leq \left[r \max_{i,j} |a_{ij}| \times n \max_{i,j} |a_{ij}| \right]^{1/2} = \sqrt{rn} \max_{i,j} |a_{ij}|.$$

For the second inequality in (17) we can write

$$\|\Phi\|_1 = \sum_{i=1}^n r_i(\Phi) \leq \sum_{i=1}^n r \max_{i,j} |a_{ij}| = rn \max_{i,j} |a_{ij}|.$$

□

Lemma 2 *The following inequality holds*

$$\int_{\mathbb{R}} \left|\phi(x; m, \sigma) - \phi(x; m', \sigma)\right| dx \leq \frac{2}{\sigma\sqrt{2\pi}} |m - m'|,$$

where $\phi(\cdot; m, \sigma)$ is the Gaussian density function with mean m and standard deviation σ.

Proof Without loss of generality we assume $m' \geq m$.

$$I = \int_{\mathbb{R}} \frac{1}{\sigma\sqrt{2\pi}} \left| \exp\left[-\frac{(x-m)^2}{2\sigma^2}\right] - \exp\left[-\frac{(x-m')^2}{2\sigma^2}\right] \right| dx = \int_{-\infty}^{\frac{m+m'}{2}} \cdots + \int_{\frac{m+m'}{2}}^{\infty} \cdots$$

$$= \int_{-\infty}^{\frac{m+m'}{2}} \frac{1}{\sigma\sqrt{2\pi}} \left[\exp\left[-\frac{(x-m)^2}{2\sigma^2}\right] - \exp\left[-\frac{(x-m')^2}{2\sigma^2}\right] \right] dx$$

$$+ \int_{\frac{m+m'}{2}}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} \left[\exp\left[-\frac{(x-m')^2}{2\sigma^2}\right] - \exp\left[-\frac{(x-m)^2}{2\sigma^2}\right] \right] dx.$$

Take $\Delta = (m' - m)/(2\sigma)$ and change the variables of the integration to get

$$\begin{aligned} I &= \int_{-\infty}^{\Delta} \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{u^2}{2}\right] du - \int_{-\infty}^{-\Delta} \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{u^2}{2}\right] du \\ &\quad + \int_{-\Delta}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{u^2}{2}\right] du - \int_{\Delta}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{u^2}{2}\right] du \\ &= 2\int_{-\Delta}^{\Delta} \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{u^2}{2}\right] du \le 2(2\Delta)\frac{1}{\sqrt{2\pi}} = \frac{2}{\sigma\sqrt{2\pi}}(m' - m). \end{aligned}$$

□

9 Appendix B

List of symbols

$\mathbb{Z} = \{0, \pm 1, \pm 2, \pm 3, \ldots\}$	The set of integers
$\mathbb{N} = \{0, 1, 2, 3, \ldots\}$	The set of non-negative integers
$\mathbb{N}_n = \{1, 2, \ldots, n\}$	The finite set of positive integers
$\mathbb{R}_{\geq 0}$	Non-negative real numbers
$\mathbb{R}^n$	n-dimensional Euclidean domain
$\mathscr{M}_{\mathfrak{s}}$	Discrete-time Markov process
$\mathcal{S}$	State space of the Markov process
$\mathcal{B}$	Borel σ-algebra on the space $\mathcal{S}$
$T_{\mathfrak{s}} : \mathcal{S} \times \mathcal{B} \to [0, 1]$	Stochastic kernel of $\mathscr{M}_{\mathfrak{s}}$
$B \in \mathcal{B}$	Any Borel set
$\Omega = \mathcal{S}^{\mathbb{N}}$	The space of trajectories of $\mathscr{M}_{\mathfrak{s}}$
$\mathcal{F}$	The product σ-algebra on Ω
$\mathcal{P}$	Probability measure on $(\Omega, \mathcal{F})$ induced by the stochastic Kernel $T_{\mathfrak{s}}$
$\boldsymbol{s}_0 \in \mathcal{S}$	Initial state of the process $\mathscr{M}_{\mathfrak{s}}$
$t_{\mathfrak{s}} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$	Conditional density function of the process $\mathscr{M}_{\mathfrak{s}}$
$\{\boldsymbol{\zeta}(t),\ t \in \mathbb{N}\}$	A sequence of iid random vectors taking values in $\mathbb{R}^n$
T_{ζ}	Distribution function of random variables $\boldsymbol{\zeta}(t),\ t \in \mathbb{N}$
r.v.	Abbreviation for random variable
$\boldsymbol{f} : \mathcal{S} \times \mathbb{R}^n \to \mathcal{S}$	A measurable map
$\boldsymbol{s} = [s_1, \ldots, s_n]^T$	Entries of the state vector $\boldsymbol{s}$
$\boldsymbol{f} = [f_1, \ldots, f_n]^T$	Entries of the map $\boldsymbol{f}$
$\boldsymbol{\zeta} = [\zeta_1, \ldots, \zeta_n]^T$	Entries of random vector $\boldsymbol{\zeta}$
$t_k : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}_{\geq 0}$	Density function obtained from the map f_k and the distribution of ζ_k.
m_i	Mass of block i in Example 2
k_i	Of the ith spring in Example 2
b_i	Coefficient of the ith damper in Example 2
$\Phi = [\Phi_{ij}]_{i,j} \in \mathbb{R}^{2n \times 2n}$	State transition matrix in Example 2
$N_{\mathfrak{d}}$	Number of heterogeneous LTI subsystems in Example 3
Φ_i	State transition matrix of the ith subsystem in Example 3
N_i	The set of subsystems to which system i is physically connected in Example 3

G_{ij}	Coefficient matrix for the effect of subsystem j to i in Example 3
B_i	Input matrix in the ith subsystem of Example 3
$A \in \mathcal{B}$	Bounded Borel set as the set of safe states
N	Finite time horizon of the invariance problem
$p_N(\boldsymbol{s}_0, A)$	Solution of the probabilistic invariance problem
$\pi : \mathcal{B} \to [0, 1]$	Probability distribution of the initial state $\boldsymbol{s}(0)$
$V_k : \mathcal{S} \to [0, 1]$	Value functions in the Bellman recursion for $\mathscr{M}_{\mathfrak{s}}$
$\mathbf{1}_A(\boldsymbol{s})$	Indicator function of the set A
$\mathfrak{B} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$	A Bayesian network (BN)
$(\mathcal{V}, \mathcal{E})$	A directed Acyclic Graph (DAG) representing the structure of $\mathfrak{B}$
$\mathcal{V}$	Nodes of the DAG representing r.v. of $\mathfrak{B}$
$\mathcal{E}$	Arcs of the DAG representing the dependence relationships among the r.v. in $\mathfrak{B}$
$\mathcal{T}$	A set containing the conditional probability distributions (CPD)
$X_i \in \mathcal{V}$	Random variables in $\mathcal{V}$
$Pa(Y)$	The parent set of the variable $Y \in \mathcal{V}$: $Pa(Y) = \{X \in \mathcal{V} \mid (X, Y) \in \mathcal{E}\}$
$\mathbb{P}(X_i \mid Pa(X_i)) \in \mathcal{T}$	Conditional probability distribution of $X_i \in \mathcal{V}$
$\mathcal{V}_1, \mathcal{V}_2$	Subsets of nodes in a two-layered BN
$X(0), X(1), X(2), \ldots$	Random variables indexed by time in a dynamic Bayesian network (DBN)
$(\mathfrak{B}_0, \mathfrak{B}_{\to})$	A DBN
$\mathfrak{B}_0$	A BN which defining the distribution of $X(0)$
$\mathfrak{B}_{\to}$	A two-layered BN defining the probability distribution for $(X(t+1) \mid X(t))$
$\mathcal{V}_1 = \{X_1, X_2, \ldots, X_n\}$	First set of random variables in the two-layered BN
$\mathcal{V}_2 = \{\bar{X}_1, \bar{X}_2, \ldots, \bar{X}_n\}$	Second set of random variables in the two-layered BN
$\boldsymbol{s}(0) = \boldsymbol{s}_0 = [s_{01}, s_{02}, \ldots, s_{0n}]^T$	Entries of the initial state $\boldsymbol{s}(0)$
$\Phi = [a_{ij}]_{i,j} \in \mathbb{R}^{n \times n}$	State matrix in the stochastic linear dynamical system
$\mathcal{N}(m, \Sigma)$	Multivariate Gaussian distribution with mean m and covariance matrix Σ
$\Sigma = diag([\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2])$	A diagonal covariance matrix
$\Pi_i : \mathbb{R}^n \to \mathbb{R}$	The projection operator defined as $\Pi_i(\boldsymbol{s}) = s_i$ for any $\boldsymbol{s} = [s_1, \ldots, s_n]^T$
$D_i \doteq \Pi_i(A)$	Projection of the safe set A over the ith dimension
$\{D_{ij}\}_{j=1}^{n_i}$	A partition for set D_i
$z_{ij} \in D_{ij}$	Representative point of the set D_{ij}
$Z_i = \{z_{i1}, \ldots, z_{in_i}\}$	Collection of the representative points in the ith dimension
ϕ_i	Absorbing state of the ith dimension
$\Omega_i = Z_i \cup \{\phi_i\}$	Finite state space of two r.v. X_i and $\bar{X}_i$
$\phi = \{\phi_1, \ldots, \phi_n\}$	The set of all absorbing states
$\Xi_i : Z_i \to 2^{D_i}$	The refinement map that associates to any point $z_{ij} \in Z_i$ the corresponding partition set $D_{ij} \subset D_i$
$\xi_i : D_i \to Z_i$	Abstraction map that associates to any point $s_i \in D_i$ the corresponding discrete state in Z_i

$T_i(\bar{X}_i \mid Pa(\bar{X}_i))$	The discrete CPD related to the node $\bar{X}_i$
$v(\cdot)$	Function that acts on a set of random variables and provides their instantiation
$z = (X_1, X_2, \ldots, X_n)$	The discrete r.v. defined over $\prod_i \Omega_i$
$Z_{\mathfrak{a}} \subset \prod_i \Omega_i$	The discrete safe set
$\mathbb{P}$	Probability measure on the event in the discrete domain
$p_N(z_0, Z_{\mathfrak{a}})$	The probabilistic invariance problem in the discrete domain
$V_k^d : \prod_i \Omega_i \to [0, 1]$	Value functions for computation of $p_N(z_0, Z_{\mathfrak{a}})$
$V_N^d(z) = \mathbf{1}_{Z_{\mathfrak{a}}}(z)$	Indicator function of the set $Z_{\mathfrak{a}}$
$\mathbb{P}(\bar{z} \mid z)$	Conditional probability distribution in the discrete domain
$\boldsymbol{j} = (j_1, j_2, \ldots, j_n)$	A vector of indices
$D_{\boldsymbol{j}} \doteq D_{1j_1} \times D_{2j_2} \times \ldots \times D_{nj_n}$	A box in the $n-$dimensional space
$\bar{A} \subset \mathbb{R}^n$	Modified safe set in the continuous domain
$W_k(\boldsymbol{s})$	Value functions for computation of the invariance probability over $\bar{A}$
$\mathcal{L}(B)$	The Lebesgue measure of any set $B \in \mathcal{B}$
$A \Delta \bar{A} \doteq (A \backslash \bar{A}) \cup (\bar{A} \backslash A)$	Symmetric difference of two sets
M	Supremum of $t_{\mathfrak{s}}(\bar{\boldsymbol{s}} \mid \boldsymbol{s})$ over $A \Delta \bar{A}$
d_{ij}	Lipschitz constant of $t_j(\bar{s}_j \mid \boldsymbol{s})$ along the ith dimension
$w_{ij} = d_{ij} \mathcal{L}(D_j)$	Weights associated to the graph of the DBN
$\mathcal{I}_j = \sum_{i=1}^n w_{ij}$	In-weight of node $\bar{X}_j$
$\mathcal{O}_i = \sum_{j=1}^n w_{ij}$	Out-weight of node X_i
$\mathcal{H}_j$	Global Lipschitz constant of $t_j(\bar{s}_j \mid \boldsymbol{s})$
$\kappa = \sum_{j=1}^n \mathcal{I}_j$	Lipschitz constant of value functions W_k
δ	Diameter of the partition for the set $\bar{A}$
δ_i	Diameter of the partition for the set D_i
ϵ	Error threshold
e	Euler's constant
$\|\Phi\|_1$	Entry-wise one-norm of matrix Φ
$\|\Phi\|_2$	Induced two-norm of matrix Φ
r	The degree of connectivity of the DAG of $\mathfrak{B}_{\rightarrow}$

References

1. Abate, A., Hillen, R.C., Wahl, S.A.: Piecewise affine approximation of fluxes and enzyme kinetics from in-vivo 13c labeling experiments. Int. J. Robust Nonlinear Control **22**(10), 1120–1139 (2012). Special Issue on System Identification for Biological Systems
2. Abate, A., Katoen, J.-P., Lygeros, J., Prandini, M.: Approximate model checking of stochastic hybrid systems. Eur. J. Control **6**, 624–641 (2010)
3. Abate, A., Katoen, J.-P., Mereacre, A.: Quantitative automata model checking of autonomous stochastic hybrid systems. In: HSCC, pp. 83–92, (2011)
4. Abate, A., Prandini, M., Lygeros, J., Sastry, S.: Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. Automatica **44**(11), 2724–2734 (2008)
5. Abate, A., Vincent, S., Dobbe, R., Silletti, A., Master, N., Axelrod, J., Tomlin, C.J.: A mechanical modeling framework for the study of epithelial morphogenesis. IEEE/ACM Trans. Comput. Biol. Bioinf. **9**(6), 1607–1620 (2012)
6. Angeles, J.: Dynamic Response of Linear Mechanical Systems—Modeling. Analysis and Simulation. Springer, New York (2012)

7. Baier, C., Katoen, J.-P.: Principles of Model Checking. MIT Press, Cambridge (2008)
8. Bertsekas, D.P.: Convergence of discretization procedures in dynamic programming. IEEE Trans. Autom. Control **20**(3), 415–419 (1975)
9. Esmaeil Zadeh Soudjani, S., Abate, A.: Higher-order approximations for verification of stochastic hybrid systems. In: ATVA, volume 7561 of LNCS, pp. 416–434. Springer, (2012)
10. Esmaeil Zadeh Soudjani, S., Abate, A.: Adaptive and sequential gridding procedures for the abstraction and verification of stochastic processes. SIAM J. Appl. Dyn. Syst. **12**(2), 921–956 (2013)
11. Esmaeil Zadeh Soudjani, S., Abate, A.: Probabilistic reach-avoid computation for partially-degenerate stochastic processes. IEEE Trans. Autom. Control **59**(2), 528–534 (2014)
12. Esmaeil Zadeh Soudjani, S., Abate, A.: Quantitative approximation of the probability distribution of a Markov process by formal abstractions. Logical methods in computer. Science **11**(3), 1–29 (2015). arXiv:1504.00039
13. Esmaeil Zadeh Soudjani, S., Abate, A., Majumdar, R.: Dynamic Bayesian networks as formal abstractions of structured stochastic processes. In: 26th International Conference on Concurrency Theory (CONCUR'15), vol. 42, pp. 169–183, (2015)
14. Esmaeil Zadeh Soudjani, S., Gevaerts, C., Abate, A.: FAUST^2: formal abstractions of uncountable-state stochastic processes. In: TACAS, volume 9035 of LNCS, pp. 272–286. Springer, (2015)
15. Gillespie, D.T.: The chemical langevin and Fokker–Planck equations for the reversible isomerization reaction. J. Phys. Chem. A **106**(20), 5063–5071 (2002)
16. Gusrialdi, A., Hirche, S.: Communication topology design for large-scale interconnected systems with time delay. In: American Control Conference, pp. 4508–4513, June (2011)
17. Jha, S.K., Clarke, E.M., Langmead, C.J., Legay, A., Platzer, A., Zuliani, P.: A Bayesian approach to model checking biological systems. In: Computational Methods in Systems Biology, volume 5688 of LNCS, pp. 218–234. Springer, (2009)
18. Katoen, J.-P., Khattri, M., Zapreev, I.S.: A Markov reward model checker. In: QEST, pp. 243–244. IEEE, (2005)
19. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques–Adaptive Computation and Machine Learning. The MIT Press, Cambridge (2009)
20. Kolotilina, L.Y.: Bounds for the singular values of a matrix involving its sparsity pattern. J. Math. Sci. **137**(3), 4794–4800 (2006)
21. Kschischang, F.R., Frey, B.J., Loeliger, H.-A.: Factor graphs and the sum-product algorithm. IEEE Trans. Inf. Theory **47**(2), 498–519 (2001)
22. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: CAV, volume 6806 of LNCS, pp. 585–591. Springer, (2011)
23. Langmead, C.J.: Generalized queries and Bayesian statistical model checking in dynamic Bayesian networks: application to personalized medicine. In: Proceedings of the 8th International Conference on Computational Systems Bioinformatics, pp. 201–212, (2009)
24. Legay, A., Delahaye, B., Bensalem, A.: Statistical Model Checking: an Overview, pp. 122–135. Springer, Berlin (2010)
25. Murphy, K.P.: Dynamic Bayesian networks: representation, inference and learning. PhD thesis, UC Berkeley, Computer Science Division, (2002)
26. Palaniappan, S.K., Thiagarajan, P.S.: Dynamic Bayesian networks: a factored model of probabilistic dynamics. In: ATVA, volume 7561 of LNCS, pp. 17–25. Springer, (2012)
27. Papoulis, A.: Probability, Random Variables, and Stochastic Processes, 3rd edn. McGraw-Hill, New York (1991)
28. Ramponi, F., Chatterjee, D., Summers, S., Lygeros, J.: On the connections between PCTL and dynamic programming. In: HSCC, pp. 253–262, (2010)
29. Sundarapandian, V.: Distributed control schemes for large-scale interconnected discrete-time linear systems. Math. Comput. Model. **41**(23), 313–319 (2005)
30. Tkachev, I., Abate, A.: Formula-free finite abstractions for linear temporal verification of stochastic hybrid systems. In: HSCC, pp. 283–292, (2013)